January 2010

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

**Preface**.................................................................................................................................................. xvii
- Documentation Accessibility ............................................................................................................ xvii
- Conventions ........................................................................................................................................ xvii

### 1 Introduction and Roadmap

1.1 Document Scope and Audience ..................................................................................................... 1-1
1.2 Guide to This Document .............................................................................................................. 1-2
1.3 Related Documentation .............................................................................................................. 1-2
1.4 New and Changed WLST Features in This Release ..................................................................... 1-3

### 2 WebLogic Server WLST Online and Offline Command Reference

2.1 WebLogic Server WLST Command Summary, Alphabetically By Command ....................... 2-1
2.2 WebLogic Server WLST Online Command Summary ............................................................. 2-6
2.3 WebLogic Server WLST Offline Command Summary ............................................................... 2-9

### 3 WLST Command and Variable Reference

3.1 Overview of WLST Command Categories ................................................................................. 3-1
3.2 Browse Commands ...................................................................................................................... 3-2
3.2.1 cd ........................................................................................................................................... 3-2
3.2.2 currentTree .......................................................................................................................... 3-3
3.2.3 prompt ................................................................................................................................. 3-4
3.2.4 pwd ....................................................................................................................................... 3-5
3.3 Control Commands ..................................................................................................................... 3-5
3.3.1 addTemplate ....................................................................................................................... 3-6
3.3.2 closeDomain ....................................................................................................................... 3-7
3.3.3 closeTemplate ..................................................................................................................... 3-7
3.3.4 connect ............................................................................................................................... 3-7
3.3.5 createDomain ..................................................................................................................... 3-10
3.3.6 disconnect ......................................................................................................................... 3-11
3.3.7 exit ....................................................................................................................................... 3-12
3.3.8 readDomain ....................................................................................................................... 3-13
3.3.9 readTemplate ..................................................................................................................... 3-13
3.3.10 updateDomain ................................................................................................................ 3-14
3.3.11 writeDomain .................................................................................................................... 3-15
3.3.12 writeTemplate ................................................................................................................ 3-16
3.4 Customization Commands ................................................................. 3-16
  3.4.1 addHelpCommandGroup ......................................................... 3-17
  3.4.2 addHelpCommand ............................................................... 3-18
3.5 Deployment Commands ................................................................. 3-18
  3.5.1 deploy ................................................................................... 3-19
  3.5.2 distributeApplication ............................................................ 3-22
  3.5.3 getWLDM ............................................................................. 3-23
  3.5.4 listApplications .................................................................. 3-24
  3.5.5 loadApplication ................................................................. 3-24
  3.5.6 redeploy ............................................................................... 3-25
  3.5.7 startApplication ................................................................. 3-26
  3.5.8 stopApplication .................................................................... 3-27
  3.5.9 undeploy ............................................................................. 3-28
  3.5.10 updateApplication ............................................................. 3-29
3.6 Diagnostics Commands ................................................................. 3-29
  3.6.1 exportDiagnosticData .......................................................... 3-30
  3.6.2 exportDiagnosticDataFromServer ....................................... 3-31
3.7 Editing Commands ........................................................................ 3-32
  3.7.1 activate ............................................................................... 3-33
  3.7.2 assign .................................................................................. 3-34
  3.7.3 cancelEdit .......................................................................... 3-36
  3.7.4 create .................................................................................. 3-37
  3.7.5 delete .................................................................................. 3-38
  3.7.6 encrypt ............................................................................... 3-39
  3.7.7 get ....................................................................................... 3-40
  3.7.8 getActivationTask .............................................................. 3-40
  3.7.9 invoke ............................................................................... 3-41
  3.7.10 isRestartRequired ............................................................. 3-42
  3.7.11 loadDB ............................................................................ 3-42
  3.7.12 loadProperties ................................................................. 3-43
  3.7.13 save .................................................................................. 3-44
  3.7.14 set ..................................................................................... 3-44
  3.7.15 setOption ......................................................................... 3-45
  3.7.16 showChanges .................................................................... 3-47
  3.7.17 startEdit ........................................................................... 3-47
  3.7.18 stopEdit ........................................................................... 3-48
  3.7.19 unassign .......................................................................... 3-49
  3.7.20 undo ................................................................................. 3-51
  3.7.21 validate ............................................................................ 3-51
3.8 Information Commands ............................................................... 3-52
  3.8.1 addListener ........................................................................ 3-53
  3.8.2 configToScript .................................................................... 3-54
  3.8.3 dumpStack .......................................................................... 3-55
  3.8.4 dumpVariables ................................................................. 3-55
  3.8.5 find ..................................................................................... 3-56
  3.8.6 getConfigManager ............................................................. 3-57
  3.8.7 getMBean ........................................................................... 3-57
<table>
<thead>
<tr>
<th>Section</th>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.8.8</td>
<td>getMBI</td>
<td>3-58</td>
</tr>
<tr>
<td>3.8.9</td>
<td>getPath</td>
<td>3-58</td>
</tr>
<tr>
<td>3.8.10</td>
<td>listChildTypes</td>
<td>3-59</td>
</tr>
<tr>
<td>3.8.11</td>
<td>lookup</td>
<td>3-59</td>
</tr>
<tr>
<td>3.8.12</td>
<td>ls</td>
<td>3-60</td>
</tr>
<tr>
<td>3.8.13</td>
<td>man</td>
<td>3-63</td>
</tr>
<tr>
<td>3.8.14</td>
<td>redirect</td>
<td>3-64</td>
</tr>
<tr>
<td>3.8.15</td>
<td>removeListener</td>
<td>3-64</td>
</tr>
<tr>
<td>3.8.16</td>
<td>showListeners</td>
<td>3-65</td>
</tr>
<tr>
<td>3.8.17</td>
<td>startRecording</td>
<td>3-65</td>
</tr>
<tr>
<td>3.8.18</td>
<td>state</td>
<td>3-66</td>
</tr>
<tr>
<td>3.8.19</td>
<td>stopRecording</td>
<td>3-67</td>
</tr>
<tr>
<td>3.8.20</td>
<td>stopRedirect</td>
<td>3-67</td>
</tr>
<tr>
<td>3.8.21</td>
<td>storeUserConfig</td>
<td>3-67</td>
</tr>
<tr>
<td>3.8.22</td>
<td>threadDump</td>
<td>3-69</td>
</tr>
<tr>
<td>3.8.23</td>
<td>viewMBean</td>
<td>3-70</td>
</tr>
<tr>
<td>3.8.24</td>
<td>writeIniFile</td>
<td>3-70</td>
</tr>
<tr>
<td>3.9</td>
<td>Life Cycle Commands</td>
<td>3-71</td>
</tr>
<tr>
<td>3.9.1</td>
<td>migrate</td>
<td>3-71</td>
</tr>
<tr>
<td>3.9.2</td>
<td>resume</td>
<td>3-72</td>
</tr>
<tr>
<td>3.9.3</td>
<td>shutdown</td>
<td>3-73</td>
</tr>
<tr>
<td>3.9.4</td>
<td>start</td>
<td>3-75</td>
</tr>
<tr>
<td>3.9.5</td>
<td>startServer</td>
<td>3-76</td>
</tr>
<tr>
<td>3.9.6</td>
<td>suspend</td>
<td>3-77</td>
</tr>
<tr>
<td>3.10</td>
<td>Node Manager Commands</td>
<td>3-78</td>
</tr>
<tr>
<td>3.10.1</td>
<td>nm</td>
<td>3-79</td>
</tr>
<tr>
<td>3.10.2</td>
<td>nmConnect</td>
<td>3-79</td>
</tr>
<tr>
<td>3.10.3</td>
<td>nmDisconnect</td>
<td>3-81</td>
</tr>
<tr>
<td>3.10.4</td>
<td>nmEnroll</td>
<td>3-82</td>
</tr>
<tr>
<td>3.10.5</td>
<td>nmGenBootStartupProps</td>
<td>3-83</td>
</tr>
<tr>
<td>3.10.6</td>
<td>nmKill</td>
<td>3-83</td>
</tr>
<tr>
<td>3.10.7</td>
<td>nmLog</td>
<td>3-84</td>
</tr>
<tr>
<td>3.10.8</td>
<td>nmServerLog</td>
<td>3-84</td>
</tr>
<tr>
<td>3.10.9</td>
<td>nmServerStatus</td>
<td>3-85</td>
</tr>
<tr>
<td>3.10.10</td>
<td>nmStart</td>
<td>3-85</td>
</tr>
<tr>
<td>3.10.11</td>
<td>nmVersion</td>
<td>3-86</td>
</tr>
<tr>
<td>3.10.12</td>
<td>startNodeManager</td>
<td>3-87</td>
</tr>
<tr>
<td>3.11</td>
<td>Tree Commands</td>
<td>3-88</td>
</tr>
<tr>
<td>3.11.1</td>
<td>custom</td>
<td>3-88</td>
</tr>
<tr>
<td>3.11.2</td>
<td>domainConfig</td>
<td>3-89</td>
</tr>
<tr>
<td>3.11.3</td>
<td>domainCustom</td>
<td>3-90</td>
</tr>
<tr>
<td>3.11.4</td>
<td>domainRuntime</td>
<td>3-91</td>
</tr>
<tr>
<td>3.11.5</td>
<td>edit</td>
<td>3-91</td>
</tr>
<tr>
<td>3.11.6</td>
<td>jndi</td>
<td>3-92</td>
</tr>
<tr>
<td>3.11.7</td>
<td>serverConfig</td>
<td>3-93</td>
</tr>
<tr>
<td>3.11.8</td>
<td>serverRuntime</td>
<td>3-93</td>
</tr>
<tr>
<td>3.12</td>
<td>WLST Variable Reference</td>
<td>3-94</td>
</tr>
</tbody>
</table>
4 Infrastructure Security Custom WLST Commands

4.1 Overview of WLST Security Commands .............................................................. 4-1
4.2 Audit Configuration Commands ........................................................................ 4-2
4.2.1 getNonJavaEEAuditMBeanName ................................................................. 4-2
4.2.2 getAuditPolicy .............................................................................................. 4-3
4.2.3 setAuditPolicy ............................................................................................... 4-3
4.2.4 getAuditRepository ....................................................................................... 4-5
4.2.5 setAuditRepository ....................................................................................... 4-5
4.2.6 listAuditEvents ............................................................................................ 4-6
4.2.7 exportAuditConfig ....................................................................................... 4-7
4.2.8 importAuditConfig ....................................................................................... 4-8
4.3 SSL Configuration Commands .......................................................................... 4-9
4.3.1 addCertificateRequest ............................................................................... 4-10
4.3.2 addSelfSignedCertificate ........................................................................... 4-10
4.3.3 changeKeyStorePassword ........................................................................... 4-11
4.3.4 changeWalletPassword .............................................................................. 4-12
4.3.5 configureSSL ............................................................................................... 4-12
4.3.6 createKeySSL ............................................................................................. 4-13
4.3.7 createWallet ................................................................................................. 4-14
4.3.8 deleteKeyStore ............................................................................................. 4-14
4.3.9 deleteWallet ................................................................................................. 4-15
4.3.10 exportKeyStore .......................................................................................... 4-15
4.3.11 exportKeyStoreObject ............................................................................... 4-16
4.3.12 exportWallet .............................................................................................. 4-17
4.3.13 exportWalletObject .................................................................................... 4-17
4.3.14 generateKey ............................................................................................... 4-19
4.3.15 getKeyStoreObject ..................................................................................... 4-19
4.3.16 getSSL ......................................................................................................... 4-20
4.3.17 getWalletObject ......................................................................................... 4-21
4.3.18 importKeyStore .......................................................................................... 4-22
4.3.19 importKeyStoreObject .............................................................................. 4-22
4.3.20 importWallet ............................................................................................. 4-23
4.3.21 importWalletObject ................................................................................... 4-24
4.3.22 listKeyStoreObjects ................................................................................... 4-25
4.3.23 listKeyStores ............................................................................................. 4-25
4.3.24 listWalletObjects ....................................................................................... 4-26
4.3.25 listWallets ................................................................................................. 4-27
4.3.26 removeKeyStoreObject .............................................................................. 4-27
4.3.27 removeWalletObject .................................................................................. 4-28
4.4 Oracle Identity Federation Commands ........................................................... 4-29
4.4.1 addConfigListEntryInMap .......................................................................... 4-31
4.4.2 addConfigMapEntryInMap ......................................................................... 4-32
4.4.3 addConfigPropertyListEntry ...................................................................... 4-32
4.4.4 addConfigPropertyMapEntry ...................................................................... 4-33
4.4.5 addCustomAuthnEngine ............................................................................ 4-33
4.4.6 addCustomSPEngine .................................................................................. 4-34
4.4.7 addFederationListEntryInMap ..................................................................... 4-34
5 Oracle WebCenter Custom WLST Commands

5.1 Oracle WebCenter WLST Command Categories ........................................ 5-2
5.2 General ................................................................. 5-2
5.2.1 deleteConnection ................................................. 5-3
5.3 Content Repository ...................................................... 5-3
5.3.1 createJCRContentServerConnection ......................... 5-4
5.3.2 setJCRContentServerConnection ................................. 5-7
5.3.3 listJCRContentServerConnections .............................. 5-9
5.3.4 createJCRPortalConnection ................................... 5-10
5.3.5 setJCRPortalConnection ........................................ 5-11
5.3.6 listJCRPortalConnections ....................................... 5-12
5.3.7 createJCRFileSystemConnection .............................. 5-13
5.3.8 setJCRFileSystemConnection .................................. 5-13
<table>
<thead>
<tr>
<th>Section</th>
<th>Method Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.7.7</td>
<td>setDefaultMailConnection</td>
<td>5-62</td>
</tr>
<tr>
<td>5.7.8</td>
<td>setMailServiceProperty</td>
<td>5-63</td>
</tr>
<tr>
<td>5.7.9</td>
<td>removeMailServiceProperty</td>
<td>5-64</td>
</tr>
<tr>
<td>5.7.10</td>
<td>listMailServiceProperties</td>
<td>5-65</td>
</tr>
<tr>
<td>5.8</td>
<td>Personal Events</td>
<td>5-65</td>
</tr>
<tr>
<td>5.8.1</td>
<td>createPersonalEventConnection</td>
<td>5-66</td>
</tr>
<tr>
<td>5.8.2</td>
<td>setPersonalEventConnection</td>
<td>5-67</td>
</tr>
<tr>
<td>5.8.3</td>
<td>listPersonalEventConnections</td>
<td>5-68</td>
</tr>
<tr>
<td>5.9</td>
<td>Portlet Producers</td>
<td>5-69</td>
</tr>
<tr>
<td>5.9.1</td>
<td>registerWSRPProducer</td>
<td>5-70</td>
</tr>
<tr>
<td>5.9.2</td>
<td>setWSRPProducer</td>
<td>5-74</td>
</tr>
<tr>
<td>5.9.3</td>
<td>listWSRPProducers</td>
<td>5-78</td>
</tr>
<tr>
<td>5.9.4</td>
<td>deregisterWSRPProducer</td>
<td>5-79</td>
</tr>
<tr>
<td>5.9.5</td>
<td>listWSRPProducerRegistrationProperties</td>
<td>5-80</td>
</tr>
<tr>
<td>5.9.6</td>
<td>listWSRPProducerUserCategories</td>
<td>5-81</td>
</tr>
<tr>
<td>5.9.7</td>
<td>mapWSRPProducerUserCategory</td>
<td>5-81</td>
</tr>
<tr>
<td>5.9.8</td>
<td>registerPDKJavaProducer</td>
<td>5-82</td>
</tr>
<tr>
<td>5.9.9</td>
<td>setPDKJavaProducer</td>
<td>5-84</td>
</tr>
<tr>
<td>5.9.10</td>
<td>deregisterPDKJavaProducer</td>
<td>5-86</td>
</tr>
<tr>
<td>5.9.11</td>
<td>listPDKJavaProducers</td>
<td>5-87</td>
</tr>
<tr>
<td>5.9.12</td>
<td>refreshProducer</td>
<td>5-88</td>
</tr>
<tr>
<td>5.9.13</td>
<td>registerOOTBProducers</td>
<td>5-88</td>
</tr>
<tr>
<td>5.9.14</td>
<td>deregisterOOTBProducers</td>
<td>5-90</td>
</tr>
<tr>
<td>5.9.15</td>
<td>registerSampleProducers</td>
<td>5-90</td>
</tr>
<tr>
<td>5.9.16</td>
<td>deregisterSampleProducers</td>
<td>5-91</td>
</tr>
<tr>
<td>5.10</td>
<td>RSS</td>
<td>5-92</td>
</tr>
<tr>
<td>5.10.1</td>
<td>getRssProxyConfig</td>
<td>5-92</td>
</tr>
<tr>
<td>5.10.2</td>
<td>setRssProxyConfig</td>
<td>5-93</td>
</tr>
<tr>
<td>5.11</td>
<td>Search</td>
<td>5-93</td>
</tr>
<tr>
<td>5.11.1</td>
<td>createSESConnection</td>
<td>5-94</td>
</tr>
<tr>
<td>5.11.2</td>
<td>setSESConnection</td>
<td>5-95</td>
</tr>
<tr>
<td>5.11.3</td>
<td>listSESConnections</td>
<td>5-96</td>
</tr>
<tr>
<td>5.11.4</td>
<td>setSearchConfig</td>
<td>5-96</td>
</tr>
<tr>
<td>5.11.5</td>
<td>setSearchSESConfig</td>
<td>5-98</td>
</tr>
<tr>
<td>5.11.6</td>
<td>listSearchConfig</td>
<td>5-98</td>
</tr>
<tr>
<td>5.11.7</td>
<td>listSearchSESConfig</td>
<td>5-99</td>
</tr>
<tr>
<td>5.12</td>
<td>WebCenter Spaces Crawlers</td>
<td>5-100</td>
</tr>
<tr>
<td>5.12.1</td>
<td>setSpacesCrawlProperties</td>
<td>5-100</td>
</tr>
<tr>
<td>5.12.2</td>
<td>getSpacesCrawlProperties</td>
<td>5-101</td>
</tr>
<tr>
<td>5.13</td>
<td>Worklists</td>
<td>5-102</td>
</tr>
<tr>
<td>5.13.1</td>
<td>createBPELConnection</td>
<td>5-102</td>
</tr>
<tr>
<td>5.13.2</td>
<td>setBPELConnection</td>
<td>5-103</td>
</tr>
<tr>
<td>5.13.3</td>
<td>listBPELConnections</td>
<td>5-104</td>
</tr>
<tr>
<td>5.13.4</td>
<td>addWorklistConnection</td>
<td>5-106</td>
</tr>
<tr>
<td>5.13.5</td>
<td>removeWorklistConnection</td>
<td>5-107</td>
</tr>
<tr>
<td>5.13.6</td>
<td>listWorklistConnections</td>
<td>5-107</td>
</tr>
<tr>
<td>5.14</td>
<td>WebCenter Spaces Workflows</td>
<td>5-108</td>
</tr>
</tbody>
</table>
6 User Messaging Service (UMS) Custom WLST Commands
   6.1 UMS WLST Command Group ................................................................. 6-1
   6.1.1 manageUserMessagingPrefs ............................................................ 6-1
   6.1.2 deployUserMessagingDriver ............................................................ 6-3

7 DMS Custom WLST Commands
   7.1 DMS Commands ................................................................................. 7-1
   7.1.1 displayMetricTableNames .............................................................. 7-1
   7.1.2 displayMetricTables ...................................................................... 7-3
   7.1.3 dumpMetrics .................................................................................. 7-6
   7.1.4 reloadMetricRules .......................................................................... 7-8

8 Logging Custom WLST Commands
   8.1 Log Configuration Commands .......................................................... 8-1
   8.1.1 configureLogHandler .................................................................... 8-2
   8.1.2 getLogLevel .................................................................................. 8-3
   8.1.3 listLoggers ................................................................................... 8-4
   8.1.4 listLogHandlers ............................................................................ 8-5
   8.1.5 setLogLevel .................................................................................. 8-6
   8.2 Search and Display Commands ......................................................... 8-7
   8.2.1 displayLogs .................................................................................. 8-7
   8.2.2 listLogs ....................................................................................... 8-10

9 Metadata Services (MDS) Custom WLST Commands
   9.1 Repository Management Commands ................................................ 9-1
   9.1.1 createMetadataPartition ............................................................. 9-2
10 Oracle SOA Suite Custom WLST Commands

10.1 Overview of WSLT Command Categories .................................................. 10-1
10.2 Deployment Commands .............................................................. 10-2
10.2.1 sca_deployComposite .................................................. 10-2
10.2.2 sca_undeployComposite ........................................ 10-4
10.3 SOA Composite Application Management Commands .......................... 10-4
10.3.1 sca_startComposite .................................................. 10-5
10.3.2 sca_stopComposite .................................................. 10-5
10.3.3 sca_activateComposite ............................................... 10-6
10.3.4 sca_retireComposite .................................................. 10-7
10.3.5 sca_assignDefaultComposite ........................................ 10-7
10.3.6 sca_listDeployedComposites ........................................ 10-8
10.4 Configuration Plan Management Commands ....................................... 10-9
10.4.1 sca_attachPlan ......................................................... 10-9
10.4.2 sca_extractPlan ......................................................... 10-10
10.4.3 sca_generatePlan ..................................................... 10-10
10.4.4 sca_validatePlan ....................................................... 10-11
10.5 Task Validation Commands .................................................................. 10-12
10.5.1 sca_validateTask ....................................................... 10-12
10.6 SOA Composite Application Compilation Commands .......................... 10-13
10.6.1 sca_setProp ............................................................. 10-13
10.6.2 sca_compile .............................................................. 10-13
10.7 SOA Composite Application Packaging Commands ............................ 10-14
10.7.1 sca_package ............................................................... 10-14
10.8 SOA Composite Application Test Commands ......................................... 10-15
10.8.1 sca_test ................................................................. 10-15
10.9 SOA Composite Application HTTP Client-Based Export and Import Commands .... 10-16
10.9.1 sca_exportComposite ................................................ 10-16
10.9.2 sca_exportUpdates .................................................... 10-18
10.9.3 sca_importUpdates .................................................... 10-19
14.2.4 setWebServiceConfiguration ................................................................. 14-7
14.2.5 listWebServiceClients .............................................................................. 14-8
14.2.6 listWebServiceClientPorts ................................................................. 14-9
14.2.7 listWebServiceClientStubProperties .................................................. 14-10
14.2.8 setWebServiceClientStubProperty ....................................................... 14-11
14.2.9 setWebServiceClientStubProperties .................................................... 14-12

15 Diagnostic Framework Custom WLST Commands
15.1 Incident Commands ................................................................................... 15-1
15.1.1 createIncident ...................................................................................... 15-2
15.1.2 getIncidentFile .................................................................................... 15-2
15.1.3 listADRHome ...................................................................................... 15-3
15.1.4 listIncidents ....................................................................................... 15-4
15.1.5 listProblems ...................................................................................... 15-4
15.1.6 showIncident ...................................................................................... 15-5
15.2 Diagnostic Dump Commands ..................................................................... 15-5
15.2.1 describeDump .................................................................................... 15-6
15.2.2 executeDump ..................................................................................... 15-6
15.2.3 listDumps ......................................................................................... 15-7

16 Information Rights Management Custom WLST Commands
16.1 Overview of WLST IRM Commands ......................................................... 16-1
16.2 General Server Commands ...................................................................... 16-2
16.2.1 addIRMRefreshPeriod ....................................................................... 16-2
16.2.2 getIRMRefreshPeriod ....................................................................... 16-3
16.2.3 getIRMRefreshPeriods ..................................................................... 16-3
16.2.4 removeIRMRefreshPeriod .................................................................. 16-4
17 Imaging and Process Management Custom WLST Commands

17.1 Overview of I/PM WLST Command Categories....................................................... 17-1
17.2 BPEL Diagnostic Commands ..................................................................................... 17-1
17.2.1 sumIPMBpelFaults............................................................................................... 17-2
17.2.2 clearIPMBpelFaults.......................................................... ............................. 17-2
17.2.3 listIPMBpelFaults.............................................................................................. 17-3
<table>
<thead>
<tr>
<th>Section</th>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>17.2.4</td>
<td>repairIPMBpelFaults</td>
<td>17-3</td>
</tr>
<tr>
<td>17.3</td>
<td>I/PM Configuration Commands</td>
<td>17-4</td>
</tr>
<tr>
<td>17.3.1</td>
<td>getIPMConfig</td>
<td>17-4</td>
</tr>
<tr>
<td>17.3.2</td>
<td>grantIPMCredAccess</td>
<td>17-5</td>
</tr>
<tr>
<td>17.3.3</td>
<td>listIPMConfig</td>
<td>17-5</td>
</tr>
<tr>
<td>17.3.4</td>
<td>refreshIPMSecurity</td>
<td>17-5</td>
</tr>
<tr>
<td>17.3.5</td>
<td>setIPMConfig</td>
<td>17-6</td>
</tr>
</tbody>
</table>
This preface describes the document accessibility features and conversions used in this guide—*WebLogic Scripting Tool Command Reference*.

**Documentation Accessibility**

Our goal is to make Oracle products, services, and supporting documentation accessible to all users, including users that are disabled. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at [http://www.oracle.com/accessibility/](http://www.oracle.com/accessibility/).

**Accessibility of Code Examples in Documentation**

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

**Accessibility of Links to External Web Sites in Documentation**

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

**Deaf/Hard of Hearing Access to Oracle Support Services**

To reach Oracle Support Services, use a telecommunications relay service (TRS) to call Oracle Support at 1.800.223.1711. An Oracle Support Services engineer will handle technical issues and provide customer support according to the Oracle service request process. Information about TRS is available at [http://www.fcc.gov/cgb/consumerfacts/trs.html](http://www.fcc.gov/cgb/consumerfacts/trs.html), and a list of phone numbers is available at [http://www.fcc.gov/cgb/dro/trsphonebk.html](http://www.fcc.gov/cgb/dro/trsphonebk.html).

**Conventions**

The following text conventions are used in this document:
<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>boldface</strong></td>
<td>Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.</td>
</tr>
<tr>
<td><em>italic</em></td>
<td>Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.</td>
</tr>
<tr>
<td><strong>monospace</strong></td>
<td>Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.</td>
</tr>
</tbody>
</table>
Introduction and Roadmap

This section describes the contents and organization of this guide—WebLogic Scripting Tool Command Reference.

- Section 1.1, "Document Scope and Audience"
- Section 1.2, "Guide to This Document"
- Section 1.3, "Related Documentation"
- Section 1.4, "New and Changed WLST Features in This Release"

1.1 Document Scope and Audience

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

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

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

1.2 Guide to This Document

This document is organized as follows:

- This chapter, "Introduction and Roadmap," introduces the organization of this guide and lists related documentation.
- Chapter 2, "WebLogic Server WLST Online and Offline Command Reference," summarizes WebLogic Server WLST commands alphabetically and by online/offline usage.
- Chapter 3, "WLST Command and Variable Reference," provides detailed descriptions for each of the WebLogic Server WLST commands and variables.
- Chapter 4, "Infrastructure Security Custom WLST Commands," provides detailed descriptions for each of the custom WLST commands that can be used to manage the Oracle Fusion Middleware Infrastructure Security components.
- Chapter 5, "Oracle WebCenter Custom WLST Commands," provides detailed descriptions for each of the custom WLST commands that can be used to manage the Oracle Fusion Middleware WebCenter component.
- Chapter 6, "User Messaging Service (UMS) Custom WLST Commands," provides detailed descriptions of the custom WLST commands that can be used to manage the Oracle Fusion Middleware User Messaging Service (UMS) component.
- Chapter 7, "DMS Custom WLST Commands," provides detailed descriptions of the custom WLST commands that can be used to manage the Oracle Fusion Middleware Dynamic Monitoring Service (DMS) component.
- Chapter 8, "Logging Custom WLST Commands," provides detailed descriptions of the custom WLST commands that can be used to manage the Oracle Fusion Middleware Logging component.
- Chapter 9, "Metadata Services (MDS) Custom WLST Commands," provides detailed descriptions of the custom WLST commands that can be used to manage the Oracle Fusion Middleware Metadata Services (MDS) component.
- Chapter 10, "Oracle SOA Suite Custom WLST Commands," provides detailed descriptions of the custom WLST commands that can be used to manage the Oracle Fusion Middleware SOA component.
- Chapter 11, "Application Development Framework (ADF) Custom WLST Commands," provides detailed descriptions of the custom WLST commands that can be used to manage the Oracle Fusion Middleware ADF component.
- Chapter 12, "Portal Custom WLST Commands," provides detailed descriptions of the custom WLST commands that can be used to manage the Oracle Fusion Middleware Portals component.
- Chapter 13, "Java Required Files Custom WLST Commands," provides detailed descriptions of the custom WLST commands that can be used to manage the Oracle Fusion Middleware JRF component.
- Chapter 14, "Web Services Custom WLST Commands," provides detailed descriptions of the custom WLST commands that can be used to manage the Oracle Fusion Middleware Web Services component.
- Chapter 15, "Diagnostic Framework Custom WLST Commands," provides detailed descriptions of the custom WLST commands that can be used to manage the Oracle Fusion Middleware Diagnostic Framework component.

1.3 Related Documentation

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

WLST is one of several interfaces for managing and monitoring WebLogic Server. For information about the other management interfaces, see:

- "Deployment Tools" in *Deploying Applications to Oracle WebLogic Server* describes several tools that WebLogic Server provides for deploying applications and stand-alone modules.
1.4 New and Changed WLST Features in This Release

The following new WLST features are available in this release:

- Support for using WLST with some Oracle Fusion Middleware products. Custom WLST commands for these products are described in this manual. These commands are WLST versions of many of the Oracle ASCTL commands for Fusion Middleware products.

- A new `domainCustom()` command that lets you access and perform operations on custom MBeans that have been registered in the Domain Runtime MBean Server. See "Accessing Custom MBeans on the Domain Runtime Server" in Oracle WebLogic Scripting Tool, and Section 3.11.3, "domainCustom" in this manual.

- Two new WLST commands, `addHelpCommandGroup()` and `addHelpCommand()`, which lets you add command group help and command help for custom WLST commands to the WLST integrated help. See Section 3.4.1, "addHelpCommandGroup" and Section 3.4.2, "addHelpCommand".

- As of this release, WLST is based on Jython 2.2.1.

For a comprehensive listing of the new WebLogic Server features introduced in this release, see What’s New in Oracle WebLogic Server.
The following sections summarize the WebLogic Server WLST commands, as follows:

- Section 2.1, "WebLogic Server WLST Command Summary, Alphabetically By Command"
- Section 2.2, "WebLogic Server WLST Online Command Summary"
- Section 2.3, "WebLogic Server WLST Offline Command Summary"

**Note:** You can list a summary of all online and offline commands from the command-line using the following commands, respectively:

```
help("online")
help("offline")
```

For information about custom WLST commands for Fusion Middleware (FMW) components, refer to the appropriate chapter in this document. For information on how to run FMW custom commands, see "Using Custom WLST Commands" in the Oracle Fusion Middleware Administrator's Guide.

## 2.1 WebLogic Server WLST Command Summary, Alphabetically By Command

The following tables summarizes each of the WebLogic Server WLST commands, alphabetically by command. This table does not include custom WLST commands for FMW components. For a list of custom commands for a given FMW component, refer to the appropriate chapter in this document.

<table>
<thead>
<tr>
<th>This command...</th>
<th>Enables you to...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>activate</td>
<td>Activate changes saved during the current editing session but not yet deployed.</td>
<td>Online</td>
</tr>
<tr>
<td>addHelpCommand</td>
<td>Adds new command help for a command to an existing command group. Once added to the group, the command (along with a brief description) is displayed in the command list for the group when you enter the help('commandGroup') command.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td>This command...</td>
<td>Enables you to...</td>
<td>Use with WLST...</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-----------------------------------------------------------------------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>addHelpCommandGroup</td>
<td>Adds a new help command group to those shown by the WLST help() command.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td>addListener</td>
<td>Add a JMX listener to the specified MBean.</td>
<td>Online</td>
</tr>
<tr>
<td>addTemplate</td>
<td>Extend the current WebLogic domain using an application or service extension template.</td>
<td>Offline</td>
</tr>
<tr>
<td>assign</td>
<td>Assign resources to one or more destinations.</td>
<td>Offline</td>
</tr>
<tr>
<td>cancelEdit</td>
<td>Cancel an edit session, release the edit lock, and discard all unsaved changes. This operation can be called by any user with administrator privileges, even if the user did not start the edit session.</td>
<td>Online</td>
</tr>
<tr>
<td>cd</td>
<td>Navigate the hierarchy of configuration or runtime beans.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td>closeDomain</td>
<td>Close the current WebLogic domain.</td>
<td>Offline</td>
</tr>
<tr>
<td>closeTemplate</td>
<td>Close the current domain template.</td>
<td>Offline</td>
</tr>
<tr>
<td>configToScript</td>
<td>Convert an existing server configuration (config directory) to an executable WLST script.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td>connect</td>
<td>Connect WLST to a WebLogic Server instance.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td>create</td>
<td>Create a configuration bean of the specified type for the current bean.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td>currentTree</td>
<td>Return the current location in the hierarchy.</td>
<td>Online</td>
</tr>
<tr>
<td>custom</td>
<td>Navigate to the root of custom MBeans that are registered in the Runtime MBean Server.</td>
<td>Online</td>
</tr>
<tr>
<td>delete</td>
<td>Delete an instance of a configuration bean of the specified type for the current configuration bean.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td>deploy</td>
<td>Deploy an application to a WebLogic Server instance.</td>
<td>Online</td>
</tr>
<tr>
<td>disconnect</td>
<td>Disconnect WLST from a WebLogic Server instance.</td>
<td>Online</td>
</tr>
<tr>
<td>distributeApplication</td>
<td>Copy the deployment bundle to the specified targets.</td>
<td>Online</td>
</tr>
<tr>
<td>domainConfig</td>
<td>Navigate to the last MBean to which you navigated in the domain configuration hierarchy or to the root of the hierarchy, DomainMBean.</td>
<td>Online</td>
</tr>
<tr>
<td>domainCustom</td>
<td>Navigate to the tree of custom MBeans that are registered in the Domain Runtime MBean Server.</td>
<td>Online</td>
</tr>
<tr>
<td>domainRuntime</td>
<td>Navigate to the last MBean to which you navigated in the domain runtime hierarchy or to the root of the hierarchy, DomainRuntimeMBean.</td>
<td>Online</td>
</tr>
<tr>
<td>dumpStack</td>
<td>Display stack trace from the last exception that occurred while performing a WLST action, and reset the stack trace.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td>dumpVariables</td>
<td>Display all variables used by WLST, including their name and value.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td>edit</td>
<td>Navigate to the last MBean to which you navigated in the configuration edit MBean hierarchy or to the root of the hierarchy, DomainMBean.</td>
<td>Online</td>
</tr>
<tr>
<td>encrypt</td>
<td>Encrypt the specified string.</td>
<td>Online</td>
</tr>
</tbody>
</table>
Table 2–1 (Cont.) WebLogic Server WLST Command Summary

<table>
<thead>
<tr>
<th>This command...</th>
<th>Enables you to...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>exit</td>
<td>Exit WLST from the user session and close the scripting shell.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td>exportDiagnosticData</td>
<td>Execute a query against the specified log file.</td>
<td>Offline</td>
</tr>
<tr>
<td>exportDiagnosticDataFromServer</td>
<td>Executes a query on the server side and retrieves the exported WebLogic Diagnostic Framework (WLDF) data.</td>
<td>Online</td>
</tr>
<tr>
<td>find</td>
<td>Find MBeans and attributes in the current hierarchy.</td>
<td>Online</td>
</tr>
<tr>
<td>get</td>
<td>Return the value of the specified attribute.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td>getActivationTask</td>
<td>Return the latest ActivationTask MBean on which a user can get status.</td>
<td>Online</td>
</tr>
<tr>
<td>getConfigManager</td>
<td>Return the latest ConfigurationManagerBean MBean which manages the change process.</td>
<td>Online</td>
</tr>
<tr>
<td>getMBean</td>
<td>Return the MBean by browsing to the specified path.</td>
<td>Online</td>
</tr>
<tr>
<td>getMBI</td>
<td>Return the MBeanInfo for the specified MBeanType or the cmo variable.</td>
<td>Online</td>
</tr>
<tr>
<td>getPath</td>
<td>Return the MBean path for the specified MBean instance.</td>
<td>Online</td>
</tr>
<tr>
<td>getWLDM</td>
<td>Return the WebLogic DeploymentManager object.</td>
<td>Online</td>
</tr>
<tr>
<td>invoke</td>
<td>Invoke a management operation on the current configuration bean.</td>
<td>Online</td>
</tr>
<tr>
<td>isRestartRequired</td>
<td>Determine whether a server restart is required.</td>
<td>Online</td>
</tr>
<tr>
<td>jndi</td>
<td>Navigates to the JNDI tree for the server to which WLST is currently connected.</td>
<td>Online</td>
</tr>
<tr>
<td>listApplications</td>
<td>List all applications that are currently deployed in the domain.</td>
<td>Online</td>
</tr>
<tr>
<td>listChildTypes</td>
<td>List all the children MBeans that can be created or deleted for the cmo.</td>
<td>Online</td>
</tr>
<tr>
<td>loadApplication</td>
<td>Load an application and deployment plan into memory.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td>loadDB</td>
<td>Load SQL files into a database.</td>
<td>Offline</td>
</tr>
<tr>
<td>loadProperties</td>
<td>Load property values from a file.</td>
<td>Online and Offline</td>
</tr>
<tr>
<td>lookup</td>
<td>Look up the specified MBean.</td>
<td>Online</td>
</tr>
<tr>
<td>ls</td>
<td>List all child beans and/or attributes for the current configuration or runtime bean.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td>man</td>
<td>Display help from MBeanInfo for the current MBean or its specified attribute.</td>
<td>Online</td>
</tr>
<tr>
<td>migrate</td>
<td>Migrate services to a target server within a cluster.</td>
<td>Online</td>
</tr>
<tr>
<td>nm</td>
<td>Determine whether WLST is connected to Node Manager.</td>
<td>Online</td>
</tr>
<tr>
<td>nmConnect</td>
<td>Connect WLST to Node Manager to establish a session.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td>Command</td>
<td>Enables you to...</td>
<td>Use with WLST...</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td><code>nmDisconnect</code></td>
<td>Disconnect WLST from a Node Manager session.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td><code>nmEnroll</code></td>
<td>Enroll the machine on which WLST is currently running.</td>
<td>Online</td>
</tr>
<tr>
<td><code>nmGenBootStartupProps</code></td>
<td>Generates the Node Manager property files, <code>boot.properties</code> and <code>startup.properties</code>, for the specified server.</td>
<td>Online</td>
</tr>
<tr>
<td><code>nmKill</code></td>
<td>Kill the specified server instance that was started with Node Manager.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td><code>nmLog</code></td>
<td>Return the Node Manager log.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td><code>nmServerLog</code></td>
<td>Return the server output log of the server that was started with Node Manager.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td><code>nmServerStatus</code></td>
<td>Return the status of the server that was started with Node Manager.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td><code>nmStart</code></td>
<td>Start a server in the current domain using Node Manager.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td><code>nmVersion</code></td>
<td>Return the Node Manager server version.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td><code>prompt</code></td>
<td>Toggle the display of path information at the prompt.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td><code>pwd</code></td>
<td>Display the current location in the configuration or runtime bean hierarchy.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td><code>readDomain</code></td>
<td>Open an existing WebLogic domain for updating.</td>
<td>Offline</td>
</tr>
<tr>
<td><code>readTemplate</code></td>
<td>Open an existing domain template for WebLogic domain creation.</td>
<td>Offline</td>
</tr>
<tr>
<td><code>redeploy</code></td>
<td>Reload classes and redeploy a previously deployed application.</td>
<td>Online</td>
</tr>
<tr>
<td><code>redirect</code></td>
<td>Redirect WLST output to the specified filename.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td><code>removeListener</code></td>
<td>Remove a listener that was previously defined.</td>
<td>Online</td>
</tr>
<tr>
<td><code>resume</code></td>
<td>Resume a server instance that is suspended or in <code>ADMIN</code> state.</td>
<td>Online</td>
</tr>
<tr>
<td><code>save</code></td>
<td>Save the edits that have been made but have not yet been saved.</td>
<td>Online</td>
</tr>
<tr>
<td><code>serverConfig</code></td>
<td>Navigate to the last MBean to which you navigated in the configuration MBean hierarchy or to the root of the hierarchy, <code>DomainMBean</code>.</td>
<td>Online</td>
</tr>
<tr>
<td><code>serverRuntime</code></td>
<td>Navigate to the last MBean to which you navigated in the runtime MBean hierarchy or to the root of the hierarchy, <code>ServerRuntimeMBean</code>.</td>
<td>Online</td>
</tr>
<tr>
<td><code>set</code></td>
<td>Set the specified attribute value for the current configuration bean.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td><code>setOption</code></td>
<td>Set options related to a WebLogic domain creation or update</td>
<td>Offline</td>
</tr>
</tbody>
</table>

Table 2–1 (Cont.) WebLogic Server WLST Command Summary
Table 2–1 (Cont.) WebLogic Server WLST Command Summary

<table>
<thead>
<tr>
<th>This command...</th>
<th>Enables you to...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>showChanges</td>
<td>Show the changes made by the current user during the current edit session.</td>
<td>Online</td>
</tr>
<tr>
<td>showListeners</td>
<td>Show all listeners that are currently defined.</td>
<td>Online</td>
</tr>
<tr>
<td>shutdown</td>
<td>Gracefully shut down a running server instance or cluster.</td>
<td>Online</td>
</tr>
<tr>
<td>start</td>
<td>Start a Managed Server instance or a cluster using Node Manager.</td>
<td>Online</td>
</tr>
<tr>
<td>startApplication</td>
<td>Start an application, making it available to users.</td>
<td>Online</td>
</tr>
<tr>
<td>startEdit</td>
<td>Start a configuration edit session on behalf of the currently connected user.</td>
<td>Online</td>
</tr>
<tr>
<td>startNodeManager</td>
<td>Start Node Manager at default port (5556).</td>
<td>Online or Offline</td>
</tr>
<tr>
<td>startRecording</td>
<td>Record all user interactions with WLST; useful for capturing commands to replay.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td>startServer</td>
<td>Start the Administration Server.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td>state</td>
<td>Returns a map of servers or clusters and their state using Node Manager.</td>
<td>Online</td>
</tr>
<tr>
<td>stopApplication</td>
<td>Stop an application, making it un available to users.</td>
<td>Online</td>
</tr>
<tr>
<td>stopEdit</td>
<td>Stop the current edit session, release the edit lock, and discard unsaved changes.</td>
<td>Online</td>
</tr>
<tr>
<td>stopRecording</td>
<td>Stop recording WLST commands.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td>stopRedirect</td>
<td>Stop the redirection of WLST output to a file.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td>storeUserConfig</td>
<td>Create a user configuration file and an associated key file.</td>
<td>Online</td>
</tr>
<tr>
<td>suspend</td>
<td>Suspend a running server.</td>
<td>Online</td>
</tr>
<tr>
<td>threadDump</td>
<td>Display a thread dump for the specified server.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td>undeploy</td>
<td>Undeploy an application from the specified servers.</td>
<td>Online</td>
</tr>
<tr>
<td>updateApplication</td>
<td>Update an application configuration using a new deployment plan.</td>
<td>Online</td>
</tr>
<tr>
<td>updateDomain</td>
<td>Update and save the current domain.</td>
<td>Offline</td>
</tr>
<tr>
<td>unassign</td>
<td>Unassign applications or services from one or more destinations.</td>
<td>Offline</td>
</tr>
<tr>
<td>undo</td>
<td>Revert all unsaved or unactivated edits.</td>
<td>Online</td>
</tr>
<tr>
<td>validate</td>
<td>Validate the changes that have been made but have not yet been saved.</td>
<td>Online</td>
</tr>
<tr>
<td>viewMBean</td>
<td>Display information about an MBean, such as the attribute names and values, and operations.</td>
<td>Online</td>
</tr>
<tr>
<td>writeDomain</td>
<td>Write the domain configuration information to the specified directory.</td>
<td>Offline</td>
</tr>
</tbody>
</table>
Table 2–2  WebLogic Server WLST Online Command Summary

<table>
<thead>
<tr>
<th>This command...</th>
<th>Enables you to...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>activate</td>
<td>Activate changes saved during the current editing session but not yet deployed.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td>addHelpCommand</td>
<td>Adds new command help for a command to an existing command group. Once added to the group, the command (along with a brief description) is displayed in the command list for the group when you enter the help('commandGroup') command.</td>
<td></td>
</tr>
<tr>
<td>addHelpCommandGroup</td>
<td>Adds a new help command group to those shown by the WLST help() command, and specifies the resource bundle in which the help information is defined for the group.</td>
<td></td>
</tr>
<tr>
<td>addListener</td>
<td>Add a JMX listener to the specified MBean.</td>
<td></td>
</tr>
<tr>
<td>cancelEdit</td>
<td>Cancel an edit session, release the edit lock, and discard all unsaved changes. This operation can be called by any user with administrator privileges, even if the user did not start the edit session.</td>
<td></td>
</tr>
<tr>
<td>cd</td>
<td>Navigate the hierarchy of configuration or runtime beans.</td>
<td></td>
</tr>
<tr>
<td>configToScript</td>
<td>Convert an existing server configuration (config directory) to an executable WLST script.</td>
<td></td>
</tr>
<tr>
<td>connect</td>
<td>Connect WLST to a WebLogic Server instance.</td>
<td></td>
</tr>
<tr>
<td>create</td>
<td>Create a configuration bean of the specified type for the current bean.</td>
<td></td>
</tr>
<tr>
<td>currentTree</td>
<td>Return the current tree location.</td>
<td></td>
</tr>
<tr>
<td>custom</td>
<td>Navigate to the root of custom MBeans that are registered in the Runtime MBean Server.</td>
<td></td>
</tr>
<tr>
<td>delete</td>
<td>Delete an instance of a configuration bean of the specified type for the current configuration bean.</td>
<td></td>
</tr>
<tr>
<td>deploy</td>
<td>Deploy an application to a WebLogic Server instance.</td>
<td></td>
</tr>
<tr>
<td>disconnect</td>
<td>Disconnect WLST from a WebLogic Server instance.</td>
<td></td>
</tr>
<tr>
<td>distributeApplication</td>
<td>Copy the deployment bundle to the specified targets.</td>
<td></td>
</tr>
<tr>
<td>domainConfig</td>
<td>Navigate to the last MBean to which you navigated in the domain configuration hierarchy or to the root of the hierarchy, DomainMBean.</td>
<td></td>
</tr>
</tbody>
</table>

2.2 WebLogic Server WLST Online Command Summary

The following table summarizes the WebLogic Server WLST online commands, alphabetically by command. This table does not include custom WLST commands for FMW components. For a list of custom commands for a given FMW component, refer to the appropriate chapter in this document.
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>domainCustom</td>
<td>Navigate to the tree of custom MBeans that are registered in the Domain Runtime MBean Server.</td>
</tr>
<tr>
<td>domainRuntime</td>
<td>Navigate to the last MBean to which you navigated in the domain runtime hierarchy or to the root of the hierarchy, DomainRuntimeMBean.</td>
</tr>
<tr>
<td>dumpStack</td>
<td>Display stack trace from the last exception that occurred, and reset the trace.</td>
</tr>
<tr>
<td>dumpVariables</td>
<td>Display all variables used by WLST, including their name and value.</td>
</tr>
<tr>
<td>edit</td>
<td>Navigate to the last MBean to which you navigated in the configuration edit MBean hierarchy or to the root of the hierarchy, DomainMBean.</td>
</tr>
<tr>
<td>encrypt</td>
<td>Encrypt the specified string.</td>
</tr>
<tr>
<td>exit</td>
<td>Exit WLST from the interactive session and close the scripting shell.</td>
</tr>
<tr>
<td>exportDiagnosticDataFromServer</td>
<td>Execute a query on the server side and retrieves the exported WebLogic Diagnostic Framework (WLDF) data.</td>
</tr>
<tr>
<td>find</td>
<td>Find MBeans and attributes in the current hierarchy.</td>
</tr>
<tr>
<td>get</td>
<td>Return the value of the specified attribute.</td>
</tr>
<tr>
<td>getActivationTask</td>
<td>Return the latest ActivationTask MBean on which a user can get status.</td>
</tr>
<tr>
<td>getConfigManager</td>
<td>Return the latest ConfigurationManagerBean MBean which manages the change process.</td>
</tr>
<tr>
<td>getMBean</td>
<td>Return the MBean by browsing to the specified path.</td>
</tr>
<tr>
<td>getMBI</td>
<td>Return the MBeanInfo for the specified MBeanType or the cmo variable.</td>
</tr>
<tr>
<td>getPath</td>
<td>Return the MBean path for the specified MBean instance.</td>
</tr>
<tr>
<td>getWLDM</td>
<td>Return the WebLogic DeploymentManager object.</td>
</tr>
<tr>
<td>invoke</td>
<td>Invoke a management operation on the current configuration bean.</td>
</tr>
<tr>
<td>isRestartRequired</td>
<td>Determine whether a server restart is required.</td>
</tr>
<tr>
<td>jndi</td>
<td>Navigates to the JNDI tree for the server to which WLST is currently connected.</td>
</tr>
<tr>
<td>listApplications</td>
<td>List all applications that are currently deployed in the domain.</td>
</tr>
<tr>
<td>listChildTypes</td>
<td>List all the children MBeans that can be created or deleted for the cmo.</td>
</tr>
<tr>
<td>loadApplication</td>
<td>Load an application and deployment plan into memory.</td>
</tr>
<tr>
<td>loadProperties</td>
<td>Load property values from a file.</td>
</tr>
<tr>
<td>lookup</td>
<td>Look up the specified MBean.</td>
</tr>
<tr>
<td>ls</td>
<td>List all child beans and/or attributes for the current configuration or runtime bean.</td>
</tr>
<tr>
<td>man</td>
<td>Display help from MBeanInfo for the current MBean or its specified attribute.</td>
</tr>
<tr>
<td>migrate</td>
<td>Migrate services to a target server within a cluster.</td>
</tr>
</tbody>
</table>
**Table 2–2** (Cont.) *WebLogic Server WLST Online Command Summary*

<table>
<thead>
<tr>
<th>This command...</th>
<th>Enables you to...</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>nm</code></td>
<td>Determine whether WLST is connected to Node Manager.</td>
</tr>
<tr>
<td><code>nmConnect</code></td>
<td>Connect WLST to Node Manager to establish a session.</td>
</tr>
<tr>
<td><code>nmDisconnect</code></td>
<td>Disconnect WLST from a Node Manager session.</td>
</tr>
<tr>
<td><code>nmEnroll</code></td>
<td>Enroll the machine on which WLST is currently running.</td>
</tr>
<tr>
<td><code>nmGenBootStartupProps</code></td>
<td>Generates the Node Manager property files, <code>boot.properties</code> and <code>startup.properties</code>, for the specified server.</td>
</tr>
<tr>
<td><code>nmKill</code></td>
<td>Kill the specified server instance that was started with Node Manager.</td>
</tr>
<tr>
<td><code>nmLog</code></td>
<td>Return the Node Manager log.</td>
</tr>
<tr>
<td><code>nmServerLog</code></td>
<td>Return the server output log of the server that was started with Node Manager.</td>
</tr>
<tr>
<td><code>nmServerStatus</code></td>
<td>Return the status of the server that was started with Node Manager.</td>
</tr>
<tr>
<td><code>nmStart</code></td>
<td>Start a server in the current domain using Node Manager.</td>
</tr>
<tr>
<td><code>nmVersion</code></td>
<td>Return the Node Manager server version.</td>
</tr>
<tr>
<td><code>prompt</code></td>
<td>Toggle the display of path information at the prompt.</td>
</tr>
<tr>
<td><code>pwd</code></td>
<td>Display the current location in the configuration or runtime bean hierarchy.</td>
</tr>
<tr>
<td><code>redeploy</code></td>
<td>Reload classes and redeploy a previously deployed application.</td>
</tr>
<tr>
<td><code>redirect</code></td>
<td>Redirect WLST output to the specified filename.</td>
</tr>
<tr>
<td><code>removeListener</code></td>
<td>Remove a listener that was previously defined.</td>
</tr>
<tr>
<td><code>resume</code></td>
<td>Resume a server instance that is suspended or in <code>ADMIN</code> state.</td>
</tr>
<tr>
<td><code>save</code></td>
<td>Save the edits that have been made but have not yet been saved.</td>
</tr>
<tr>
<td><code>serverConfig</code></td>
<td>Navigate to the last MBean to which you navigated in the configuration MBean hierarchy or to the root of the hierarchy, <code>DomainMBean</code>.</td>
</tr>
<tr>
<td><code>serverRuntime</code></td>
<td>Navigate to the last MBean to which you navigated in the runtime MBean hierarchy or to the root of the hierarchy, <code>ServerRuntimeMBean</code>.</td>
</tr>
<tr>
<td><code>set</code></td>
<td>Set the specified attribute value for the current configuration bean.</td>
</tr>
<tr>
<td><code>showChanges</code></td>
<td>Show the changes made by the current user during the current edit session.</td>
</tr>
<tr>
<td><code>showListeners</code></td>
<td>Show all listeners that are currently defined.</td>
</tr>
<tr>
<td><code>shutdown</code></td>
<td>Gracefully shut down a running server instance or cluster.</td>
</tr>
<tr>
<td><code>start</code></td>
<td>Start a Managed Server instance or a cluster using Node Manager.</td>
</tr>
<tr>
<td><code>startApplication</code></td>
<td>Start an application, making it available to users.</td>
</tr>
<tr>
<td><code>startEdit</code></td>
<td>Start a configuration edit session on behalf of the currently connected user.</td>
</tr>
<tr>
<td><code>startNodeManager</code></td>
<td>Start Node Manager at default port (5556).</td>
</tr>
</tbody>
</table>
2.3 WebLogic Server WLST Offline Command Summary

The following table summarizes the WebLogic Server WLST offline commands, alphabetically by command.

<table>
<thead>
<tr>
<th>Command</th>
<th>Enables you to...</th>
</tr>
</thead>
<tbody>
<tr>
<td>addHelpCommand</td>
<td>Adds new command help for a command to an existing command group. Once added to the group, the command (along with a brief description) is displayed in the command list for the group when you enter the help('commandGroup') command.</td>
</tr>
<tr>
<td>addHelpCommandGroup</td>
<td>Adds a new help command group to those shown by the WLST help() command, and specifies the resource bundle in which the help information is defined for the group.</td>
</tr>
<tr>
<td>addTemplate</td>
<td>Extend the current domain using an application or service extension template.</td>
</tr>
<tr>
<td>assign</td>
<td>Assign resources to one or more destinations.</td>
</tr>
<tr>
<td>cd</td>
<td>Navigate the hierarchy of configuration or runtime beans.</td>
</tr>
<tr>
<td>closeDomain</td>
<td>Close the current domain.</td>
</tr>
<tr>
<td>closeTemplate</td>
<td>Close the current domain template.</td>
</tr>
<tr>
<td>startRecording</td>
<td>Record all user interactions with WLST; useful for capturing commands to replay.</td>
</tr>
<tr>
<td>startServer</td>
<td>Start the Administration Server.</td>
</tr>
<tr>
<td>state</td>
<td>Returns a map of servers or clusters and their state using Node Manager.</td>
</tr>
<tr>
<td>stopApplication</td>
<td>Stop an application, making it unavailable to users.</td>
</tr>
<tr>
<td>stopEdit</td>
<td>Stop the current edit session, release the edit lock, and discard unsaved changes.</td>
</tr>
<tr>
<td>stopRecording</td>
<td>Stop recording WLST commands.</td>
</tr>
<tr>
<td>stopRedirect</td>
<td>Stop the redirection of WLST output to a file.</td>
</tr>
<tr>
<td>storeUserConfig</td>
<td>Create a user configuration file and an associated key file.</td>
</tr>
<tr>
<td>suspend</td>
<td>Suspend a running server.</td>
</tr>
<tr>
<td>threadDump</td>
<td>Display a thread dump for the specified server.</td>
</tr>
<tr>
<td>undeploy</td>
<td>Undeploy an application from the specified servers.</td>
</tr>
<tr>
<td>undo</td>
<td>Revert all unsaved or unactivated edits.</td>
</tr>
<tr>
<td>updateApplication</td>
<td>Update an application configuration using a new deployment plan.</td>
</tr>
<tr>
<td>validate</td>
<td>Validate the changes that have been made but have not yet been saved.</td>
</tr>
<tr>
<td>viewMBean</td>
<td>Display information about an MBean, such as the attribute names and values, and operations.</td>
</tr>
<tr>
<td>writeIniFile</td>
<td>Convert WLST definitions and method declarations to a Python (.py) file.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>configToScript</td>
<td>Convert an existing server configuration (config directory) to an executable WLST script.</td>
</tr>
<tr>
<td>connect</td>
<td>Connect WLST to a WebLogic Server instance.</td>
</tr>
<tr>
<td>create</td>
<td>Create a configuration bean of the specified type for the current bean.</td>
</tr>
<tr>
<td>delete</td>
<td>Delete an instance of a configuration bean of the specified type for the current configuration bean.</td>
</tr>
<tr>
<td>dumpStack</td>
<td>Display stack trace from the last exception that occurred while performing a WLST action, and reset the stack trace.</td>
</tr>
<tr>
<td>dumpVariables</td>
<td>Display all variables used by WLST, including their name and value.</td>
</tr>
<tr>
<td>exit</td>
<td>Exit WLST from the interactive session and close the scripting shell.</td>
</tr>
<tr>
<td>exportDiagnosticData</td>
<td>Execute a query against the specified log file.</td>
</tr>
<tr>
<td>get</td>
<td>Return the value of the specified attribute.</td>
</tr>
<tr>
<td>loadDB</td>
<td>Load SQL files into a database.</td>
</tr>
<tr>
<td>loadProperties</td>
<td>Load property values from a file.</td>
</tr>
<tr>
<td>ls</td>
<td>List all child beans and/or attributes for the current configuration or runtime bean.</td>
</tr>
<tr>
<td>nmConnect</td>
<td>Connect WLST to Node Manager to establish a session.</td>
</tr>
<tr>
<td>prompt</td>
<td>Toggle the display of path information at the prompt.</td>
</tr>
<tr>
<td>pwd</td>
<td>Display the current location in the configuration or runtime bean hierarchy.</td>
</tr>
<tr>
<td>readDomain</td>
<td>Open an existing WebLogic domain for updating.</td>
</tr>
<tr>
<td>readTemplate</td>
<td>Open an existing domain template for domain creation.</td>
</tr>
<tr>
<td>redirect</td>
<td>Redirect WLST output to the specified filename.</td>
</tr>
<tr>
<td>set</td>
<td>Set the specified attribute value for the current configuration bean.</td>
</tr>
<tr>
<td>setOption</td>
<td>Set options related to a WebLogic domain creation or update.</td>
</tr>
<tr>
<td>startNodeManager</td>
<td>Start Node Manager at default port (5556).</td>
</tr>
<tr>
<td>startRecording</td>
<td>Record all user interactions with WLST; useful for capturing commands to replay.</td>
</tr>
<tr>
<td>startServer</td>
<td>Start the Administration Server.</td>
</tr>
<tr>
<td>stopRecording</td>
<td>Stop recording WLST commands.</td>
</tr>
<tr>
<td>stopRedirect</td>
<td>Stop the redirection of WLST output to a file.</td>
</tr>
<tr>
<td>threadDump</td>
<td>Display a thread dump for the specified server.</td>
</tr>
<tr>
<td>unassign</td>
<td>Unassign applications or services from one or more destinations.</td>
</tr>
<tr>
<td>updateDomain</td>
<td>Update and save the current domain.</td>
</tr>
<tr>
<td>writeDomain</td>
<td>Write the domain configuration information to the specified directory.</td>
</tr>
<tr>
<td>writeIniFile</td>
<td>Convert WLST definitions and method declarations to a Python (.py) file.</td>
</tr>
<tr>
<td>Command</td>
<td>Enables you to</td>
</tr>
<tr>
<td>-----------</td>
<td>----------------------------------------------------</td>
</tr>
<tr>
<td>writeTemplate</td>
<td>Writes the domain configuration information to the specified domain template.</td>
</tr>
</tbody>
</table>
The following sections describe the WLST commands and variables in detail. Topics include:

- Section 3.1, "Overview of WLST Command Categories"
- Section 3.2, "Browse Commands"
- Section 3.3, "Control Commands"
- Section 3.4, "Customization Commands"
- Section 3.5, "Deployment Commands"
- Section 3.6, "Diagnostics Commands"
- Section 3.7, "Editing Commands"
- Section 3.8, "Information Commands"
- Section 3.9, "Life Cycle Commands"
- Section 3.10, "Node Manager Commands"
- Section 3.11, "Tree Commands"
- Section 3.12, "WLST Variable Reference"

### 3.1 Overview of WLST Command Categories

**Note:** It is recommended that you review "Syntax for WLST Commands" in Oracle WebLogic Scripting Tool for command syntax requirements.

WLST commands are divided into the following categories.

<table>
<thead>
<tr>
<th>Command Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Section 3.2, &quot;Browse Commands&quot;</td>
<td>Navigate the hierarchy of configuration or runtime beans and control the prompt display.</td>
</tr>
</tbody>
</table>
| Section 3.3, "Control Commands" |  - Connect to or disconnect from a server.  
  - Create and configure a WebLogic domain or domain template.  
  - Exit WLST. |
3.2 Browse Commands

Use the WLST browse commands, listed in Table 3–2, to navigate the hierarchy of configuration or runtime beans and control the prompt display.

Table 3–2  Browse Commands for WLST Configuration

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>cd</td>
<td>Navigate the hierarchy of configuration or runtime beans.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td>currentTree</td>
<td>Return the current location in the hierarchy.</td>
<td>Online</td>
</tr>
<tr>
<td>prompt</td>
<td>Toggle the display of path information at the prompt.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td>pwd</td>
<td>Display the current location in the hierarchy.</td>
<td>Online or Offline</td>
</tr>
</tbody>
</table>

3.2.1 cd

Command Category: Browse Commands

Use with WLST: Online or Offline

3.2.1.1 Description

Navigates the hierarchy of configuration or runtime beans. This command uses a model that is similar to navigating a file system in a Windows or UNIX command shell. For example, to navigate back to a parent configuration or runtime bean, enter cd ('..'). The character string ' .. ' (dot-dot), refers to the directory immediately...
above the current directory. To get back to the root bean after navigating to a bean that is deep in the hierarchy, enter \texttt{cd(’/’)}.

You can navigate to beans in the current hierarchy and to any child or instance.

The \texttt{cd} command returns a stub of the configuration or runtime bean instance, if one exists. If you navigate to a type, this command returns a stub of the configuration or runtime bean instance from which you navigated. In the event of an error, the command returns a \texttt{WLSTException}.

\begin{center}
\textbf{Note:} The \texttt{cmo} variable is initialized to the root of all domain configuration beans when you first connect WLST to a server instance. It reflects the parent configuration bean type until you navigate to an instance. For more information about the \texttt{cmo} variable, see "Changing the Current Management Object" in \textit{Oracle WebLogic Scripting Tool}.
\end{center}

### 3.2.1.2 Syntax

\texttt{cd(mbeanName)}

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{mbeanName}</td>
<td>Path to the bean in the namespace.</td>
</tr>
</tbody>
</table>

### 3.2.1.3 Examples

The following example navigates the hierarchy of configuration beans. The first command navigates to the \texttt{Servers} configuration bean type, the second, to the \texttt{myserver} configuration bean instance, and the last back up two levels to the original directory location.

\begin{verbatim}
wls:/mydomain/serverConfig> cd('Servers')
wls:/mydomain/serverConfig/Servers> cd('myserver')
wls:/mydomain/serverConfig/Servers/myserver> cd('..../..')
wls:/mydomain/serverConfig>
\end{verbatim}

### 3.2.2 currentTree

Command Category: Browse Commands

Use with WLST: Online

#### 3.2.2.1 Description

Returns the current location in the hierarchy. This command enables you to store the current location in the hierarchy and easily return to it after browsing. In the event of an error, the command returns a \texttt{WLSTException}.

#### 3.2.2.2 Syntax

\texttt{currentTree()}

#### 3.2.2.3 Example

The following example stores the current location in the hierarchy in \texttt{myTree} and uses it to navigate back to the Edit MBean hierarchy from the runtime MBean hierarchy on an Administration Server instance.

\begin{verbatim}
wls:/mydomain/edit> myTree=currentTree() wls:/mydomain/edit> serverRuntime()
\end{verbatim}
Location changed to serverRuntime tree. This is a read-only tree with ServerRuntimeMBean as the root.
For more help, use help('serverRuntime')

wls:/mydomain/serverRuntime> myTree()
wls:/mydomain/edit>

3.2.3 prompt

Command Category: Browse Commands
Use with WLST: Online or Offline

3.2.3.1 Description
Toggles the display of path information at the prompt, when entered without an argument. This command is useful when the prompt becomes too long due to the length of the path.

You can also explicitly specify on or off as an argument to the command. When you specify off, WLST hides the WLST prompt and defaults to the Jython prompt. By default, the WLST prompt displays the configuration or runtime navigation path information.

When you disable the prompt details, to determine your current location in the hierarchy, you can use the pwd command, as described in Section 3.2.4, "pwd".

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

3.2.3.2 Syntax
prompt(myPrompt)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>myPrompt</td>
<td>Optional. Hides or displays WLST prompt. Valid values include off or on.</td>
</tr>
<tr>
<td></td>
<td>■ The off argument hides the WLST prompt.</td>
</tr>
<tr>
<td></td>
<td>If you run prompt('off'), when using WLST online, the prompt defaults to the Jython prompt. You can create a new prompt using Jython syntax. For more information about programming using Jython, see <a href="http://www.jython.org">http://www.jython.org</a>. In this case, if you subsequently enter the prompt command without arguments, WLST displays the WLST command prompt without the path information. To redisplay the path information, enter prompt() again, or enter prompt('on').</td>
</tr>
<tr>
<td></td>
<td>■ The on argument displays the default WLST prompt, including the path information.</td>
</tr>
</tbody>
</table>

3.2.3.3 Examples
The following example hides and then redisplays the path information at the prompt.

wls:/mydomain/serverConfig/Servers/myserver> prompt()
wls://> prompt()
wls:/mydomain/serverConfig/Servers/myserver>

The following example hides the prompt and defaults to the Jython prompt (since the command is run using WLST online), changes the Jython prompt, and then redisplays the WLST prompt. This example also demonstrates the use of the pwd command.
3.2.4 pwd

Command Category: Browse Commands
Use with WLST: Online or Offline

3.2.4.1 Description
Displays the current location in the configuration or runtime bean hierarchy.
This command is useful when you have turned off the prompt display of the path information using the `prompt` command, as described in Section 3.2.3, "prompt".
In the event of an error, the command returns a WLSTException.

3.2.4.2 Syntax
`pwd()`

3.2.4.3 Example
The following example displays the current location in the configuration bean hierarchy.

```
wlst:/mydomain/serverConfig/Servers/myserver> prompt('off')
>>>sys.ps1="myprompt>"
myprompt> prompt()
wls:> pwd()
'serverConfig:Servers/myserver'
wls:> prompt()
wls:/mydomain/serverConfig/Servers/myserver>
```

```
wls:/mydomain/serverConfig/Servers/myserver> pwd()
'serverConfig:/Servers/myserver/Log/myserver'
```

3.3 Control Commands

Use the WLST control commands, listed in Table 3–3, to perform the following tasks:
- Connect to or disconnect from a server (connect and disconnect commands)
- Create a new WebLogic domain from a domain template, similar to the Configuration Wizard (createDomain, readTemplate, writeDomain, and closeTemplate commands)
- Update an existing WebLogic domain, offline (readDomain, addTemplate, updateDomain, and closeDomain commands)
- Write a domain template (writeTemplate command)
- Exit WLST

Table 3–3 lists the control commands for WLST configuration.

---

**Note:** For more information about programming using Jython, see [http://www.jython.org](http://www.jython.org).
### 3.3.1 addTemplate

**Command Category:** Control Commands  
**Use with WLST:** Offline

#### 3.3.1.1 Description

Extends the current WebLogic domain using an application or service extension template. Use the Template Builder to create an application or service extension template. See Oracle WebLogic Server Creating Templates Using the Domain Template Builder.

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

#### 3.3.1.2 Syntax

```
addTemplate(templateFileName)
```

#### 3.3.1.3 Example

The following example opens a WebLogic domain and extends it using the specified extension template, `DefaultWebApp.jar`.

```
wls:/offline> readDomain('c:/Oracle/Middleware/user_projects/domains/wlw')
```

---

### Table 3–3  Control Commands for WLST Configuration

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>connect</td>
<td>Connect WLST to a WebLogic Server instance.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td>disconnect</td>
<td>Disconnect WLST from a WebLogic Server instance.</td>
<td>Online</td>
</tr>
<tr>
<td>createDomain</td>
<td>Create a new WebLogic domain using the specified template.</td>
<td>Offline</td>
</tr>
<tr>
<td>readTemplate</td>
<td>Open an existing domain template for domain creation.</td>
<td>Offline</td>
</tr>
<tr>
<td>writeDomain</td>
<td>Write the domain configuration information to the specified directory.</td>
<td>Offline</td>
</tr>
<tr>
<td>closeTemplate</td>
<td>Close the current domain template.</td>
<td>Offline</td>
</tr>
<tr>
<td>readDomain</td>
<td>Open an existing WebLogic domain for updating.</td>
<td>Offline</td>
</tr>
<tr>
<td>addTemplate</td>
<td>Extend the current WebLogic domain using an application or service extension template.</td>
<td>Offline</td>
</tr>
<tr>
<td>updateDomain</td>
<td>Update and save the current domain.</td>
<td>Offline</td>
</tr>
<tr>
<td>closeDomain</td>
<td>Close the current domain.</td>
<td>Offline</td>
</tr>
<tr>
<td>writeTemplate</td>
<td>Writes the configuration information to the specified domain template file.</td>
<td>Offline</td>
</tr>
<tr>
<td>exit</td>
<td>Exit WLST from the interactive session and close the scripting shell.</td>
<td>Online or Offline</td>
</tr>
</tbody>
</table>

---

**Argument Definition**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>templateFileName</td>
<td>Name of the application or service extension template.</td>
</tr>
</tbody>
</table>
3.3.2 closeDomain

Command Category: Control Commands
Use with WLST: Offline

3.3.2.1 Description
Closes the current domain. The domain is no longer available for editing once it is closed. In the event of an error, the command returns a WLSTException.

3.3.2.2 Syntax
closeDomain()

3.3.2.3 Example
The following example closes the current domain:

```
ws:/offline/wlw> readDomain('c:/Oracle/Middleware/user_projects/domains/medrec')
...
ws:/offline/medrec> updateDomain()
ws:/offline/medrec> closeDomain()
ws:/offline>
```

3.3.3 closeTemplate

Command Category: Control Commands
Use with WLST: Offline

3.3.3.1 Description
Closes the current domain template. The domain template is no longer available once it is closed. In the event of an error, the command returns a WLSTException.

3.3.3.2 Syntax
closeTemplate()

3.3.3.3 Example
The following example opens an existing domain template, performs some operations, and then closes the current domain template.

```
ws:/offline> readTemplate('c:/Oracle/Middleware/wlserver_10.3/common/templates/domains/wls.jar')
...
ws:/offline/wls> closeTemplate()
ws:/offline>
```

3.3.4 connect

Command Category: Control Commands
Use with WLST: Online or Offline

```
ws:/offline/wlw> addTemplate('c:/Oracle/Middleware/wlserver_10.3/common/templates/applications/DefaultWebApp.jar')
ws:/offline/wlw>
```
3.3.4.1 Description

Connects WLST to a WebLogic Server instance.

Requires you to provide the credentials (user name and password) of a user who has been defined in the active WebLogic security realm. Once you are connected, a collection of security policies determine which configuration attributes you are permitted to view or modify. (See "Default Security Policies for MBeans" in the WebLogic Server MBean Reference.)

You can supply user credentials by doing any of the following:

- Enter the credentials on the command line. This option is recommended only if you are using WLST in interactive mode.
- Enter the credentials on the command line, then use the storeUserConfig command to create a user configuration file that contains your credentials in an encrypted form and a key file that WebLogic Server uses to unencrypt the credentials. On subsequent WLST sessions (or in WLST scripts), supply the name of the user configuration file and key file instead of entering the credentials on the command line. This option is recommended if you use WLST in script mode because it prevents you from storing unencrypted user credentials in your scripts.
- Use the credentials that are stored in the Administration Server's boot.properties file. By default, when you create an Administration Server, WebLogic Server encrypts the credentials used to create the server and stores them in a boot.properties file.

If you run the connect command without specifying the username and password or user configuration file and key file, WLST attempts to process the command using one of the methods listed below (in order of precedence):

1. If a user configuration and default key file exists in your home directory, then use those files. The location of the home directory depends on the type of operating system on which WLST is running. For information about the default location, see Section 3.8.21, "storeUserConfig".
2. If the adminServerName argument is not specified, then look for the boot.properties file in ./boot.properties or ./servers/myserver/security/boot.properties.
3. If the adminServerName argument is specified, then look for the boot.properties file in ./servers/adminServerName/security/boot.properties, where adminServerName is the value of the adminServerName argument.

Please note:

- Oracle strongly recommends that you connect WLST to the server through the SSL port or administration port. If you do not, the following warning message is displayed:

Warning: An insecure protocol was used to connect to the server. To ensure on-the-wire security, the SSL port or Admin port should be used instead.

- If you are connecting to a WebLogic Server instance through an SSL listen port on a server that is using the demonstration SSL keys and certificates, invoke WLST using the following command:

  java -Dweblogic.security.SSL.ignoreHostnameVerification=true -Dweblogic.security.TrustKeyStore=DemoTrust weblogic.WLST
For more information about invoking WLST, see "Main Steps for Using WLST in Interactive or Script Mode" in Oracle WebLogic Scripting Tool.

- If you are connecting to a WebLogic Server instance via HTTP, ensure that the TunnelingEnabled attribute is set to true for the WebLogic Server instance. For more information, see "TunnelingEnabled" in Oracle WebLogic Server MBean Reference.

After successfully connecting to a WebLogic Server instance, all the local variables are initialized.

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

### 3.3.4.2 Syntax

```plaintext
connect([username, password], [url], [timeout])
connect([userConfigFile, userKeyFile], [url], [timeout])
connect([url], [adminServerName], [timeout])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>Optional. Username of the operator who is connecting WLST to the server. If not specified, WLST processes the command as described above.</td>
</tr>
<tr>
<td>password</td>
<td>Optional. Password of the operator who is connecting WLST to the server. If not specified, WLST processes the command as described above.</td>
</tr>
<tr>
<td>url</td>
<td>Optional. Listen address and listen port of the server instance, specified using the following format: <code>[protocol://]listen-address:listen-port</code>. If not specified, this argument defaults to <code>t3://localhost:7001</code>.</td>
</tr>
<tr>
<td>timeout</td>
<td>Optional. The number of milliseconds that WLST waits for online commands to complete (return).</td>
</tr>
<tr>
<td></td>
<td>When you invoke a WLST online command, WLST connects to an MBean Server, invokes an MBean server method, and returns the results of the invocation. If the MBean server method does not return within the timeout period, WLST abandons its invocation attempt. Use the following syntax for this argument: <code>timeout='milliseconds'</code></td>
</tr>
<tr>
<td></td>
<td>A value of 0 indicates that the operation will not time out. This argument defaults to 300,000 ms (or 5 minutes).</td>
</tr>
<tr>
<td>userConfigFile</td>
<td>Optional. Name and location of a user configuration file which contains an encrypted username and password. Use the following syntax for this argument: <code>userConfigFile='file-system-path'</code></td>
</tr>
<tr>
<td></td>
<td>If not specified, WLST processes the command as described above.</td>
</tr>
<tr>
<td></td>
<td>When you create a user configuration file, the storeUserConfig command uses a key file to encrypt the username and password. Only the key file that encrypts a user configuration file can decrypt the username and password. (See Section 3.8.21, &quot;storeUserConfig&quot;.)</td>
</tr>
<tr>
<td>userKeyFile</td>
<td>Optional. Name and location of the key file that is associated with the specified user configuration file and is used to decrypt it. Use the following syntax for this argument: <code>userKeyFile='file-system-path'</code></td>
</tr>
<tr>
<td></td>
<td>If not specified, WLST processes the command as described above.</td>
</tr>
<tr>
<td></td>
<td>See Section 3.8.21, &quot;storeUserConfig&quot;.</td>
</tr>
</tbody>
</table>
3.3.4.3 Examples

The following example connects WLST to a WebLogic Server instance. In this example, the Administration Server name defaults to AdminServer. Note that a warning is displayed if the SSL or administration port is not used to connect to the server.

```
wlst:/offline> connect('weblogic','welcome1','t3://localhost:8001')
Connecting to weblogic server instance running at t3://localhost:8001 as username weblogic...
Successfully connected to Admin Server 'AdminServer' that belongs to domain 'mydomain'.
```

Warning: An insecure protocol was used to connect to the server. To ensure on-the-wire security, the SSL port or Admin port should be used instead.

```
wlst:/mydomain/serverConfig>
```

The following example connects WLST to a WebLogic Server instance at the specified URL. In this example, the username and password are passed as variables. This example uses a secure protocol.

```
wls:/offline> username = 'weblogic'
wls:/offline> password = 'welcome1'
wls:/offline> connect(username,password,'t3s://myhost:8001')
Connecting to weblogic server instance running at t3://myhost:8001 as username weblogic...
Successfully connected to Admin Server 'AdminServer' that belongs to domain 'mydomain'.
wls:/mydomain/serverConfig>
```

The following example connects WLST to a WebLogic Server instance using a user configuration and key file to provide user credentials.

```
wls:/offline> connect(userConfigFile='c:/myfiles/myuserconfigfile.secure',
userKeyFile='c:/myfiles/myuserkeyfile.secure')
Connecting to weblogic server instance running at t3://localhost:7001 as username ...
Successfully connected to Admin Server 'AdminServer' that belongs to domain 'mydomain'.
wls:/mydomain/serverConfig>
```

3.3.5 createDomain

Command Category: Control Commands
Use with WLST: Offline

### 3.3.5.1 Description
Creates a WebLogic domain using the specified template.

**Note:** If you wish to modify the domain configuration settings when creating a WebLogic domain, see Option 2 in "Editing a Domain (Offline)" in *Oracle WebLogic Scripting Tool*.

The `createDomain` command is similar in functionality to the `unpack` command, as described in *Creating Templates and Domains Using the pack and unpack Commands*.

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

### 3.3.5.2 Syntax
```python
createDomain(domainTemplate, domainDir, user, password)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>domainTemplate</td>
<td>Name and location of the domain template from which you want to create a domain.</td>
</tr>
<tr>
<td>domainDir</td>
<td>Name of the directory to which you want to write the domain configuration information.</td>
</tr>
<tr>
<td>user</td>
<td>Name of the default user.</td>
</tr>
<tr>
<td>password</td>
<td>Password of the default user.</td>
</tr>
</tbody>
</table>

### 3.3.5.3 Example
The following example creates a new WebLogic domain using the Avitek MedRec template and sets the default username to `weblogic` and the password to `welome1`. The domain is saved to the following directory:

```
c:/Oracle/Middleware/wlserver_10.3/user_projects/domains/medrec
```

```
wls:/offline> createDomain('c:/Oracle/Middleware/wlserver_10.3/common/templates/domains/wls_medrec.jar','c:/Oracle/Middleware/user_projects/domains/medrec','weblogic', 'welome1')
```

### 3.3.6 disconnect

Command Category: Control Commands

Use with WLST: Online

#### 3.3.6.1 Description
Disconnects WLST from a WebLogic Server instance. The `disconnect` command does not cause WLST to exit the interactive scripting shell; it closes the current WebLogic Server instance connection and resets all the variables while keeping the interactive shell alive.

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

You can connect to another WebLogic Server instance using the `connect` command, as described in *Section 3.3.4, "connect"*. 
3.3.6.2 Syntax

disconnect(force)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>force</td>
<td>Optional. Boolean value specifying whether WLST should disconnect without waiting for the active sessions to complete. This argument defaults to false, indicating that all active sessions must complete before disconnect.</td>
</tr>
</tbody>
</table>

3.3.6.3 Example

The following example disconnects from a running server:

wls:/mydomain/serverConfig> disconnect()
Disconnected from weblogic server: myserver
wls:/offline>

3.3.7 exit

Command Category: Control Commands
Use with WLST: Online or Offline

3.3.7.1 Description

Exits WLST from the user session and closes the scripting shell.

If there is an edit session in progress, WLST prompts you for confirmation. To skip the prompt, set the defaultAnswer argument to y.

By default, WLST calls System.exit(0) for the current WLST JVM when exiting WLST. If you would like the JVM to exit with a different exit code, you can specify a value using the exitCode argument.

Note: When the WLST exit command is issued within an Ant script, it may also exit the execution of the Ant script. It is recommended that when invoking WLST within an Ant script, you fork a new JVM by specifying fork="true".

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

3.3.7.2 Syntax

exit([defaultAnswer], [exitcode])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>defaultAnswer</td>
<td>Optional. Default response, if you would prefer not to be prompted at the command line. Valid values are y and n. This argument defaults to null, and WLST prompts you for a response.</td>
</tr>
<tr>
<td>exitcode</td>
<td>Optional. Exit code to set when exiting WLST.</td>
</tr>
</tbody>
</table>

3.3.7.3 Example

The following example disconnects from the user session and closes the scripting shell.

wls:/mydomain/serverConfig> exit()
Exiting WebLogic Scripting Tool...
c:\>

The following example disconnects from the user session, closes the scripting shell, and sets the error code to 101.

```
wlst:/mydomain/serverConfig> exit(exitcode=101)
Exiting WebLogic Scripting Tool ... 
```
c:\>

### 3.3.8 readDomain

**Command Category:** Control Commands  
**Use with WLST:** Offline

#### 3.3.8.1 Description

Opens an existing WebLogic domain for updating.

WLST offline provides read and write access to the configuration data that is persisted in the `config` directory for the WebLogic domain, or in a domain template JAR created using Template Builder. This data is a collection of XML documents and expresses a hierarchy of management objects.

When you open a template or WebLogic domain, WLST is placed at the root of the configuration hierarchy for that domain, and the prompt is updated to reflect the current location in the configuration hierarchy. For example:

```
wls:/offline/base_domain>
```

For more information, see "Navigating and Interrogating MBeans" in *Oracle WebLogic Scripting Tool*.

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

#### 3.3.8.2 Syntax

```
readDomain(domainDirName)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>domainDirName</td>
<td>Name of the WebLogic domain directory that you want to open.</td>
</tr>
</tbody>
</table>

#### 3.3.8.3 Example

The following example opens the `medrec` domain for editing.

```
wls:/offline> readDomain('c:/Oracle/Middleware/user_projects/domains/medrec')
wls:/offline/medrec>
```

### 3.3.9 readTemplate

**Command Category:** Control Commands  
**Use with WLST:** Offline

#### 3.3.9.1 Description

Opens an existing domain template for domain creation.
When you open a domain template, WLST is placed into the configuration bean hierarchy for that domain template, and the prompt is updated to reflect the current location in the configuration hierarchy. For example:

```
wlst:/offline/base_domain>
```

WebLogic Server configuration beans exist within a hierarchical structure. In the WLST file system, the hierarchies correspond to drives; types and instances are directories; attributes and operations are files. WLST traverses the hierarchical structure of configuration beans using commands such as `cd`, `ls`, and `pwd` in a similar way that you would navigate a file system in a UNIX or Windows command shell. After navigating to a configuration bean instance, you interact with the bean using WLST commands. For more information, see "Navigating and Interrogating MBeans" in Oracle WebLogic Scripting Tool.

**Note:** Using WLST and a domain template, you can only create and access security information when you are creating a new WebLogic domain. When you are updating a WebLogic domain, you cannot access security information through WLST.

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

### 3.3.9.2 Syntax

```
readTemplate(templateFileName)
```

**Argument Definition**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>templateFileName</td>
<td>Name of the JAR file corresponding to the domain template.</td>
</tr>
</tbody>
</table>

### 3.3.9.3 Example

The following example opens the `medrec.jar` domain template for WebLogic domain creation.

```
wls:/offline> readTemplate('c:/Oracle/Middleware/wlserver_10.3/common/templates/domains/wls_medrec.jar')
wls:/offline/wls_medrec>
```

### 3.3.10 updateDomain

**Command Category:** Control Commands

**Use with WLST:** Offline

#### 3.3.10.1 Description

Updates and saves the current WebLogic domain. The domain continues to be editable after you update and save it.

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

#### 3.3.10.2 Syntax

```
updateDomain()
```
3.3.10.3 Example
The following examples opens the medrec domain, performs some operations, and updates and saves the current domain:

```bash
wls:/offline> readDomain('c:/Oracle/Middleware/user_projects/domains/medrec')
...
wls:/offline/medrec> updateDomain()
```

3.3.11 writeDomain

Command Category: Control Commands
Use with WLST: Offline

3.3.11.1 Description
Writes the domain configuration information to the specified directory.

Once you write the WebLogic domain to file system, you can continue to update the domain template object that exists in memory, and reissue the `writeDomain` command to store the domain configuration to a new or existing file.

By default, when you write a WebLogic domain, the associated applications are written to `WL_HOME/user_projects/applications/domainname`, where `WL_HOME` specifies the WebLogic Server home directory and `domainname` specifies the name of the WebLogic domain. This directory must be empty; otherwise, WLST displays an error.

When you have finished using the domain template object in memory, close it using the `closeTemplate` command. If you want to edit the WebLogic domain that has been saved to disk, you can open it using the `readDomain` command.

---

### Note:
The name of the WebLogic domain is derived from the name of the domain directory. For example, for a domain saved to `c:/Oracle/Middleware/user_projects/domains/myMedrec`, the domain name is `myMedrec`.

---

Before writing the domain, you must define a password for the default user, if it is not already defined. For example:

```bash
cd('/Security/base_domain/User/weblogic')
cmo.setPassword('welcome1')
```

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

3.3.11.2 Syntax

```bash
writeDomain(domainDir)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>domainDir</td>
<td>Name of the directory to which you want to write the domain configuration information.</td>
</tr>
</tbody>
</table>

3.3.11.3 Example

The following example reads the medrec.jar domain templates, performs some operations, and writes the domain configuration information to the `c:/Oracle/Middleware/user_projects/domains/medrec` directory.
3.3.12 writeTemplate

Command Category: Control Commands
Use with WLST: Offline

3.3.12.1 Description
Writes the domain configuration information to the specified domain template. You can use the domain configuration template to recreate the WebLogic domain.

Once your write the configuration information to the domain configuration template, you can continue to update the WebLogic domain or domain template object that exists in memory, and reissue the writeDomain or writeTemplate command to store the domain configuration to a new or existing WebLogic domain or domain template file. For more information, see Section 3.3.11, "writeDomain" or Section 3.3.12, "writeTemplate", respectively.

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

---

**Note:** The writeTemplate command is similar in functionality to the pack command; see "The pack Command" in Creating Templates and Domains Using the pack and unpack Commands. However, writeTemplate does not support creating a Managed Server template.

---

3.3.12.2 Syntax
writeTemplate(templateName)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>templateName</code></td>
<td>Name of the domain template to store the domain configuration information.</td>
</tr>
</tbody>
</table>

3.3.12.3 Example
The following example writes the current domain configuration to the domain template named `c:/Oracle/Middleware/user_projects/templates/myTemplate.jar`.

```
wls:/offline> readTemplate('c:/Oracle/Middleware/wlsserver_10.3/common/templates/domains/wls.jar')
...
wls:/offline/base_domain> writeDomain('c:/Oracle/Middleware/user_projects/domains/base_domain')
```

3.4 Customization Commands

Use the WLST customization commands, listed in Table 3–4, to add the command group help and command help that is listed by the WLST help() and help('commandGroup') commands. For more information about adding command
help to WLST, see “Adding Integrated Help for Custom Commands” in Oracle WebLogic Scripting Tool.

### Table 3–4 Customization Commands for WLST Configuration

<table>
<thead>
<tr>
<th>This command...</th>
<th>Enables you to...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>addHelpCommandGroup</td>
<td>Adds a new help command group to those shown by the WLST help() command.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td>addHelpCommand</td>
<td>Adds new command help for a command to an existing command group. Once added to the group, the command (along with a brief description) is displayed in the command list for the group when you enter the help('commandGroup') command.</td>
<td>Online or Offline</td>
</tr>
</tbody>
</table>

### 3.4.1 addHelpCommandGroup

Command Category: Customization Commands

Use with WLST: Online or Offline

#### 3.4.1.1 Description

Adds a new command help group to those shown by the WLST help() command, and specifies the resource bundle in which the help information is defined for the group.

#### 3.4.1.2 Syntax

`addHelpCommandGroup(commandGroup, resourceBundleName)`

**Argument Definition**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>commandGroup</td>
<td>Use a unique name for the command group. Do not use a command group name that is already shown by the WLST help() command.</td>
</tr>
<tr>
<td>resourceBundleName</td>
<td>Represents either a class name or property resource file name. The resource bundle contains help text for entries for the command group using a standard pattern. The resource bundle name will be passed to ResourceBundle.getBundle(...). Multiple command groups can use the same resource bundle. The resource bundle must be present in the classpath. See “Adding Integrated Help for Custom Commands” in Oracle WebLogic Scripting Tool for information on how to define the help text for each command group and command. For more information on resourceBundles and localization, refer to <a href="http://java.sun.com/javase/6/docs/api/java/util/ResourceBundle.html">http://java.sun.com/javase/6/docs/api/java/util/ResourceBundle.html</a>.</td>
</tr>
</tbody>
</table>

#### 3.4.1.3 Examples

The following example adds the `boot` command group to the list of groups shown by the help() command, and specifies that the help text is located in the property resource file `myhelp`:

```
wlst:/offline> addHelpCommandGroup('boot', 'myhelp')
```
The following example adds the `boot` command group to the list of groups shown by the `help()` command, and specifies that the help text is located in the class `foo.bar.MyResourceBundleClass`:

```
  wls:/offline> addHelpCommandGroup('boot','foo.bar.MyResourceBundleClass')
```

### 3.4.2 addHelpCommand

Command Category: Customization Commands

Use with WLST: Online or Offline

#### 3.4.2.1 Description

Adds new command help for a command to an existing command group. Once added to the group, the command (along with a brief description) is displayed in the command list for the group when you enter the `help('commandGroup')` command. You can also specify whether or not the command is listed by the `help('online')` and `help('offline')` commands.

#### 3.4.2.2 Syntax

```
  addHelpCommand(commandName,commandGroup,[offline=false, online=false])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>commandName</td>
<td>The name of the command as defined in the command group specified by commandGroup.</td>
</tr>
<tr>
<td>commandGroup</td>
<td>The commandGroup to which the command belongs.</td>
</tr>
<tr>
<td>online</td>
<td>Optional. Boolean value that determines whether or not the command shows up in the <code>help('online')</code> output. The default value is <code>'false'</code>.</td>
</tr>
<tr>
<td>offline</td>
<td>Optional. Boolean value that determines whether or not the command shows up in the <code>help('offline')</code> output. The default value is <code>'false'</code>.</td>
</tr>
</tbody>
</table>

#### 3.4.2.3 Example

The following example shows how to add the online command `bootDB` to the listing output by the `help('boot')` and `help('online')` commands:

```
  wls:/offline> addHelpCommand('bootDB','boot',online='true',offline='false')
```

### 3.5 Deployment Commands

Use the WLST deployment commands, listed in Table 3–5, to:

- Deploy, undeploy, and redeploy applications and standalone modules to a WebLogic Server instance.
- Update an existing deployment plan.
- Interrogate the WebLogic Deployment Manager object.
- Start and stop a deployed application.

For more information about deploying applications, see *Deploying Applications to Oracle WebLogic Server*. 

---

**Table 3–5:** Deployment Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>deploy</td>
<td>Deploy the application to the WebLogic Server.</td>
</tr>
<tr>
<td>undeploy</td>
<td>Undeploy the application from the WebLogic Server.</td>
</tr>
<tr>
<td>redeploy</td>
<td>Redeploy the application to the WebLogic Server.</td>
</tr>
<tr>
<td>update</td>
<td>Update the deployment plan.</td>
</tr>
<tr>
<td>start</td>
<td>Start the application.</td>
</tr>
<tr>
<td>stop</td>
<td>Stop the application.</td>
</tr>
</tbody>
</table>
## 3.5.1 `deploy`

**Command Category:** Deployment Commands  
**Use with WLST:** Online

### Description
Deploys an application to a WebLogic Server instance.

The `deploy` command returns a `WLSTProgress` object that you can access to check the status of the command. For more information about the `WLSTProgress` object, see "WLSTProgress Object" in Oracle WebLogic Scripting Tool. In the event of an error, the command returns a `WLSTException`.

**Note:** If there is an edit session in progress, the `deploy` command does not block user interaction.

### Syntax
```
deploy(appName, path, [targets], [stageMode], [planPath], [options])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>appName</code></td>
<td>Name of the application or standalone Java EE module to be deployed.</td>
</tr>
<tr>
<td><code>path</code></td>
<td>Name of the application directory, archive file, or root of the exploded archive directory to be deployed.</td>
</tr>
<tr>
<td>Argument</td>
<td>Definition</td>
</tr>
<tr>
<td>------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>targets</strong></td>
<td>Optional. Comma-separated list of the targets. Each target may be qualified with a Java EE module name (for example, <code>module1@server1</code>) enabling you to deploy different modules of the application archive on different servers. This argument defaults to the server to which WLST is currently connected.</td>
</tr>
<tr>
<td><strong>stageMode</strong></td>
<td>Optional. Staging mode for the application you are deploying. Valid values are <code>stage</code>, <code>nostage</code>, and <code>external_stage</code>. For information about the staging modes, see &quot;Controlling Deployment File Copying with Staging Modes&quot; in Deploying Applications to Oracle WebLogic Server. This argument defaults to null.</td>
</tr>
<tr>
<td><strong>planPath</strong></td>
<td>Optional. Name of the deployment plan file. The filename can be absolute or relative to the application directory. This argument defaults to the <code>plan/plan.xml</code> file in the application directory, if one exists.</td>
</tr>
<tr>
<td><strong>options</strong></td>
<td>Optional. Comma-separated list of deployment options, specified as name-value pairs. Valid options include:</td>
</tr>
<tr>
<td></td>
<td>- <strong>altDD</strong>—Location of the alternate application deployment descriptor on the Administration Server.</td>
</tr>
<tr>
<td></td>
<td>- <strong>altWlsDD</strong>—Location of the alternate WebLogic application deployment descriptor on the Administration Server.</td>
</tr>
<tr>
<td></td>
<td>- <strong>archiveVersion</strong>—Archive version number.</td>
</tr>
<tr>
<td></td>
<td>- <strong>block</strong>—Boolean value specifying whether WLST should block user interaction until the command completes. This option defaults to <code>true</code>. If set to <code>false</code>, WLST returns control to the user after issuing the command; you can query the <code>WLSTProgress</code> object to determine the status of the command. If you are importing WLST as a Jython module, as described in &quot;Importing WLST as a Jython Module&quot; in Oracle WebLogic Scripting Tool, <code>block</code> is always set to <code>true</code>.</td>
</tr>
<tr>
<td></td>
<td>- <strong>clusterDeploymentTimeout</strong>—Time, in milliseconds, granted for a cluster deployment task on this application.</td>
</tr>
<tr>
<td></td>
<td>- <strong>createPlan</strong>—Boolean value indicating that user would like to create a default plan. This option defaults to <code>false</code>.</td>
</tr>
<tr>
<td></td>
<td>- <strong>defaultSubmoduleTargets</strong>—Boolean value indicating that targeting for qualifying JMS submodules should be derived by the system, see &quot;Using Sub-Module Targeting with JMS Application Modules&quot; in Deploying Applications to Oracle WebLogic Server. Default value is <code>true</code>.</td>
</tr>
<tr>
<td></td>
<td>- <strong>deploymentPrincipalName</strong>—String value specifying the principal for deploying the file or archive during server starts (static deployment; it does not effect the current deployment task). Make sure the user exists. This option adds <code>&lt;deployment-principal-name&gt;</code> to the <code>&lt;app-deployment&gt;</code> element in the <code>config.xml</code> file.</td>
</tr>
<tr>
<td></td>
<td>- <strong>forceUndeployTimeout</strong>—Force undeployment timeout value.</td>
</tr>
<tr>
<td></td>
<td>- <strong>gracefulIgnoreSessions</strong>—Boolean value specifying whether the graceful production to admin mode operation should ignore pending HTTP sessions. This option defaults to <code>false</code> and only applies if <code>gracefulProductionToAdmin</code> is set to <code>true</code>.</td>
</tr>
<tr>
<td></td>
<td>- <strong>gracefulProductionToAdmin</strong>—Boolean value specifying whether the production to Admin mode operation should be graceful. This option defaults to <code>false</code>.</td>
</tr>
<tr>
<td></td>
<td>- <strong>libImplVersion</strong>—Implementation version of the library, if it is not present in the manifest.</td>
</tr>
<tr>
<td></td>
<td>- <strong>libraryModule</strong>—Boolean value specifying whether the module is a library module. This option defaults to <code>false</code>.</td>
</tr>
</tbody>
</table>
3.5.1.3 Example

The following example deploys the businessApp application located at c:/myapps/business. A default deployment plan is created.

The deploy command returns a WLSTProgress object that you can access to check the status of the command. The WLSTProgress object is captured in a user-defined variable, in this case, progress.

wls:/mydomain/serverConfig/Servers> progress= deploy(appName='businessApp', path='c:/myapps/business',createplan='true')

The previous example stores the WLSTProgress object returned in a user-defined variable, in this case, progress. You can then use the progress variable to print the status of the deploy command. For example:

wls:/mydomain/serverConfig/Servers> progress.printStatus()

Current Status of your Deployment:
Deployment command type: deploy
Deployment State : completed
Deployment Message : null

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>options</td>
<td>Contains additional parameters for the deployment. (Continued)</td>
</tr>
<tr>
<td></td>
<td>- libSpecVersion—Specification version of the library, if it is not present in the manifest.</td>
</tr>
<tr>
<td></td>
<td>- planVersion—Plan version number.</td>
</tr>
<tr>
<td></td>
<td>- remote—Boolean value specifying whether the operation will be remote from the file system that contains the source. Use this option when you are on a different machine from the Administration Server and the deployment files are already at the specified location where the Administration Server is located. This option defaults to false.</td>
</tr>
<tr>
<td></td>
<td>- retireGracefully—Retirement policy to gracefully retire an application only after it has completed all in-flight work. This policy is only meaningful for stop and redeploy operations and is mutually exclusive to the retire timeout policy.</td>
</tr>
<tr>
<td></td>
<td>- retireTimeout—Time (in seconds) WLST waits before retiring an application that has been replaced with a newer version. This option default to -1, which specifies graceful timeout.</td>
</tr>
<tr>
<td></td>
<td>- securityModel—Security model. Valid values include: DDOnly, CustomRoles, CustomRolesAndPolicies, and Advanced.</td>
</tr>
<tr>
<td></td>
<td>- securityValidationEnabled—Boolean value specifying whether security validation is enabled.</td>
</tr>
<tr>
<td></td>
<td>- subModuleTargets—Submodule level targets for JMS modules. For example, <a href="mailto:submod@mod-jms.xml">submod@mod-jms.xml</a>@target</td>
</tr>
<tr>
<td></td>
<td>- testMode—Boolean value specifying whether to start the Web application with restricted access. This option defaults to false.</td>
</tr>
<tr>
<td></td>
<td>- timeout—Time (in milliseconds) that WLST waits for the deployment process to complete before canceling the operation. A value of 0 indicates that the operation will not time out. This argument defaults to 300,000 ms (or 5 minutes).</td>
</tr>
<tr>
<td></td>
<td>- upload—Boolean value specifying whether the application files are uploaded to the WebLogic Server Administration Server's upload directory prior to deployment. Use this option when the Administration Server cannot access the application files through the file system. This option defaults to false.</td>
</tr>
<tr>
<td></td>
<td>- versionIdentifier—Version identifier.</td>
</tr>
</tbody>
</table>
For more information about the WLSTProgress object, see “WLSTProgress Object” in Oracle WebLogic Scripting Tool.

The following example deploys the demoApp application in the archive file located at c:/myapps/demos/app/demoApp.ear, targeting the application modules to myserver, and using the deployment plan file located in c:/myapps/demos/app/plan/plan.xml. WLST waits 120,000 ms for the process to complete.

```
wlst:/mydomain/serverConfig/Servers> deploy('demoApp',
'c:/myapps/demos/app/demoApp.ear', targets='myserver',
planPath='c:/myapps/demos/app/plan/plan.xml', timeout=120000)
```

The following example deploys the jmsApp application located at c:/myapps/demos/jmsApp/demo-jms.xml, targeting the application module to a specific target.

```
wlst:/mydomain/serverConfig/Servers> deploy('jmsApp',path=
'c:/myapps/demos/jmsApps/demo-jms.xml', subModuleTargets='jmsApp@managed1')
```

The following example shows how to set the application version (appVersion) to a unique identifier to support production (side-by-side) redeployment. This example deploys the demoApp application in the archive file located at c:/myapps/demos/app/demoApp.ear, and sets the application and archive version numbers to the specified values.

```
wlst:/mydomain/serverConfig> deploy('demoApp', 'c:/myapps/demos/app/demoApp.ear',
archiveVersion='901-101', appVersion='901-102')
```

For more information about production redeployment strategies, see “Redeploying Applications in a Production Environment” in Deploying Applications to Oracle WebLogic Server.

### 3.5.2 distributeApplication

Command Category: Deployment Commands

Use with WLST: Online

#### 3.5.2.1 Description

Copies the deployment bundle to the specified targets. The deployment bundle includes module, configuration data, and any additional generated code. The distributeApplication command does not start deployment.

The distributeApplication command returns a WLSTProgress object that you can access to check the status of the command. For more information about the WLSTProgress object, see “WLSTProgress Object” in Oracle WebLogic Scripting Tool. In the event of an error, the command returns a WLSTException.

#### 3.5.2.2 Syntax

```
distributeApplication(appPath, [planPath], [targets], [options])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appPath</td>
<td>Name of the archive file or root of the exploded archive directory to be deployed.</td>
</tr>
</tbody>
</table>
3.5.2.3 Example

The following example loads the BigApp application located in the \(c:/myapps\) directory, and stores the WLSTProgress object in a user-defined variable, in this case, \(progress\).

\[
\text{wls:/offline> progress=distributeApplication('c:/myapps/BigApp', 'c:/deployment/BigApp/plan.xml', 'myserver,oamserver1,oamcluster')}
\]

Distributing Application and Plan ...
Successfully distributed the application.

The previous example stores the WLSTProgress object in a user-defined variable, in this case, progress. You can then use the \(progress\) variable to determine if the distributeApplication command has completed. For example:

\[
\text{wls:/mydomain/serverConfig/Servers> progress.isCompleted()}
1
\]

For more information about the WLSTProgress object, see "WLSTProgress Object" in Oracle WebLogic Scripting Tool.

3.5.3 getWLDM

Command Category: Deployment Commands

Use with WLST: Online

3.5.3.1 Description

Returns the WebLogic DeploymentManager object. You can use the object methods to configure and deploy applications. WLST must be connected to an Administration Server to run this command. In the event of an error, the command returns a WLSTException.

3.5.3.2 Syntax

getWLDM()
3.5.4 listApplications

Command Category: Deployment Commands

Use with WLST: Online

3.5.4.1 Description
Lists all applications that are currently deployed in the WebLogic domain.

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

3.5.4.2 Syntax
listApplications()

3.5.4.3 Example
The following example lists all the applications currently deployed in mydomain.

wls:/mydomain/serverConfig> listApplications()
SamplesSearchWebApp
asyncServletEar
jspSimpleTagEar
ejb30
webservicesJwsSimpleEar
ejb20BeanMgedEar
xmlBeanEar
extServletAnnotationsEar
examplesWebApp
apache_xbean.jar
mainWebApp
jdbcRowSetsEar

3.5.5 loadApplication

Command Category: Deployment Commands

Use with WLST: Online

3.5.5.1 Description
Loads an application and deployment plan into memory.

The loadApplication command returns a WLSTPlan object that you can access to make changes to the deployment plan. For more information about the WLSTPlan object, see "WLSTPlan Object" in Oracle WebLogic Scripting Tool. In the event of an error, the command returns a WLSTException.

3.5.5.2 Syntax
loadApplication(appPath, [planPath], [createPlan])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appPath</td>
<td>Name of the top-level parent application directory, archive file, or root of the exploded archive directory containing the application to be loaded.</td>
</tr>
</tbody>
</table>
### 3.5.5.3 Example

The following example loads the `c:/myapps/myejb.jar` application using the plan file at `c:/myplans/myejb/plan.xml`.

```
wlst:/myserver/serverConfig> myPlan=loadApplication('c:/myapps/myejb.jar', 'c:/myplans/myejb/plan.xml')
Loading application from c:/myapps/myejb.jar and deployment plan from c:/myplans/myejb/plan.xml ...
Successfully loaded the application.
wlst:/myserver/serverConfig>
```

The previous example stores the `WLSTPlan` object returned in the `myPlan` variable. You can then use `myPlan` variable to display information about the plan, such as the variables. For example:

```
wls:/myserver/serverConfig> myPlan.showVariables()
MyEJB jndi.ejb
MyWAR app.foo
wls:/myserver/serverConfig>
```

For more information about the `WLSTPlan` object, see "WLSTPlan Object" in *Oracle WebLogic Scripting Tool*.

### 3.5.6 redeploy

Command Category: Deployment Commands

Use with WLST: Online

#### 3.5.6.1 Description

Reloads classes and redeploy a previously deployed application.

The `redeploy` command returns a `WLSTProgress` object that you can access to check the status of the command. For more information about the `WLSTProgress` object, see "WLSTProgress Object" in *Oracle WebLogic Scripting Tool*.

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

For more information about redeploying applications, see "Overview of Common Deployment Scenarios" in *Deploying Applications to Oracle WebLogic Server*.

#### 3.5.6.2 Syntax

```
redeploy(appName, [planPath], [options])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the application to be redeployed.</td>
</tr>
</tbody>
</table>

### Argument Definition

- **planPath**: Optional. Name of the deployment plan file. The filename can be absolute or relative to the application directory. This argument defaults to the `plan/plan.xml` file in the application directory, if one exists.

- **createPlan**: Optional. Boolean value specifying whether WLST should create a plan in the application directory if the specified plan does not exist. This argument defaults to `true`. 

---

**Additional Note:**

- **WLSTPlan Object**
- **WLSTProgress Object**
- **WLSTException**
- **Deploying Applications to Oracle WebLogic Server**
- *Oracle WebLogic Scripting Tool*
3.5.6.3 Example

The following example redeploys myApp application using the plan.xml file located in the c:/myapps directory.

```plaintext
wls:/mydomain/serverConfig> progress=redeploy('myApp' 'c:/myapps/plan.xml')
Redeploying application 'myApp' ...
Redeployment of 'myApp' is successful
wls:/mydomain/serverConfig>
```

The previous example stores the WLSTProgress object returned in a user-defined variable, in this case, progress. You can then use the progress variable to access the state of the redeploy command. For example:

```plaintext
wls:/mydomain/serverConfig/Servers> progress.getState()
'completed'
wls:/mydomain/serverConfig/Servers>
```

For more information about the WLSTProgress object, see “WLSTProgress Object” in Oracle WebLogic Scripting Tool.

3.5.7 startApplication

Command Category: Deployment Commands

Use with WLST: Online

3.5.7.1 Description

Starts an application, making it available to users. The application must be fully configured and available in the WebLogic domain.

The startApplication command returns a WLSTProgress object that you can access to check the status of the command. For more information about the WLSTProgress object, see “WLSTProgress Object” in Oracle WebLogic Scripting Tool. In the event of an error, the command returns a WLSTException.

3.5.7.2 Syntax

```plaintext
startApplication(appName, [options])
```

### Argument Definition

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>planPath</strong></td>
<td>Optional. Name of the deployment plan file. The filename can be absolute or relative to the application directory. This argument defaults to the plan/plan.xml file in the application directory, if one exists.</td>
</tr>
</tbody>
</table>
| **options**  | Optional. Comma-separated list of deployment options, specified as name-value pairs. For a list of valid deployment options, see options argument description in Section 3.5.1, “deploy”. In addition, the following deployment option can be specified for the redeploy command:  
  - **appPath**—Name of the archive file or root of the exploded archive directory to be redeployed.  
  - **deploymentPrincipalName**—String value specifying the principal for redeploying the file or archive during server starts. You can use this option to overwrite the current `<deployment-principal-name>` in the config.xml file. |
3.5.7.3 Example

The following example starts the BigApp application with the specified deployment options.

```
wls:/mydomain/serverConfig/Servers> progress=startApplication('BigApp', stageMode='NOSTAGE', testMode='false')
Starting the application...
Successfully started the application.
```

The previous example stores the WLSTProgress object returned in a user-defined variable, in this case, `progress`. You can then use the `progress` variable to access the state of the `startApplication` command. For example:

```
wls:/mydomain/serverConfig/Servers> progress.getState()
'completed'
wls:/mydomain/serverConfig/Servers>
```

For more information about the WLSTProgress object, see "WLSTProgress Object" in Oracle WebLogic Scripting Tool.

3.5.8 stopApplication

Command Category: Deployment Commands

Use with WLST: Online

3.5.8.1 Description

Stops an application, making it unavailable to users. The application must be fully configured and available in the WebLogic domain.

The `stopApplication` command returns a WLSTProgress object that you can access to check the status of the command. For more information about the WLSTProgress object, see "WLSTProgress Object" in Oracle WebLogic Scripting Tool.

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

3.5.8.2 Syntax

```
stopApplication(appName, [options])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the application to stop, as specified in the plan.xml file.</td>
</tr>
<tr>
<td>options</td>
<td>Optional. Comma-separated list of deployment options, specified as name-value pairs. For a list of valid deployment options, see <code>options</code> argument description in Section 3.5.1, &quot;deploy&quot;.</td>
</tr>
</tbody>
</table>

3.5.8.3 Example

The following example stops the BigApp application.

```
wls:/offline> progress=stopApplication('BigApp')
```
Stopping the application...
Successfully stopped the application.

The previous example stores the WLSTProgress object returned in a user-defined variable, in this case, progress. You can then use the progress variable to check whether stopApplication command is running. For example:

```plaintext
wls:/mydomain/serverConfig/Servers> progress.isRunning()
0
wls:/mydomain/serverConfig/Servers>
```

For more information about the WLSTProgress object, see “WLSTProgress Object” in Oracle WebLogic Scripting Tool.

### 3.5.9 undeploy

Command Category: Deployment Commands

Use with WLST: Online

#### 3.5.9.1 Description
Undeploys an application from the specified servers.

The undeploy command returns a WLSTProgress object that you can access to check the status of the command. For more information about the WLSTProgress object, see “WLSTProgress Object” in Oracle WebLogic Scripting Tool. In the event of an error, the command returns a WLSTException.

For more information about deploying and undeploying applications, see “Overview of Common Deployment Scenarios” in Deploying Applications to Oracle WebLogic Server.

#### 3.5.9.2 Syntax

```
undeploy(appName,[targets],[options])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Deployment name for the deployed application.</td>
</tr>
<tr>
<td>targets</td>
<td>Optional. List of the target servers from which the application will be removed. If not specified, defaults to all current targets.</td>
</tr>
<tr>
<td>options</td>
<td>Optional. Comma-separated list of deployment options, specified as name-value pairs. For a list of valid deployment options, see options argument description in Section 3.5.1, “deploy”.</td>
</tr>
</tbody>
</table>

#### 3.5.9.3 Example

The following example removes the businessApp application from all target servers. WLST waits 60,000 ms for the process to complete.

```plaintext
wls:/mydomain/serverConfig> undeploy('businessApp', timeout=60000)
Undeploying application businessApp ...
Completed the undeployment of Application with status
Current Status of your Deployment:
  Deployment command type: undeploy
  Deployment State : completed
  Deployment Message : no message
```
3.5.10 updateApplication

Command Category: Deployment Commands

Use with WLST: Online

3.5.10.1 Description

Updates an application configuration using a new deployment plan. The application
must be fully configured and available in the WebLogic domain.

The `updateApplication` command returns a `WLSTProgress` object that you can
access to check the status of the command. For more information about the
`WLSTProgress` object, see "WLSTProgress Object" in `Oracle WebLogic Scripting Tool`. In
the event of an error, the command returns a `WLSTException`.

3.5.10.2 Syntax

`updateApplication(appName, [planPath], [options])`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the application, as specified in the current <code>plan.xml</code> file.</td>
</tr>
<tr>
<td>planPath</td>
<td>Optional. Name of the new deployment plan file. The filename can be absolute or relative to the application directory.</td>
</tr>
<tr>
<td>options</td>
<td>Optional. Comma-separated list of deployment options, specified as name-value pairs. For a list of valid deployment options, see <code>options</code> argument description in Section 3.5.1, &quot;deploy&quot;.</td>
</tr>
</tbody>
</table>

3.5.10.3 Example

The following example updates the application configuration for `BigApp` using the `plan.xml` file located in `c:/myapps/BigApp/newPlan`.

```
ws:/offline> progress=updateApplication('BigApp',
                    'c:/myapps/BigApp/newPlan/plan.xml', stageMode='STAGE', testMode='false')
Updating the application...
Successfully updated the application.
```

The previous example stores the `WLSTProgress` object returned in a user-defined variable, in this case, `progress`. You can then use the `progress` variable to access the state of the `updateApplication` command. For example:

```plaintext
ws:/mydomain/serverConfig/Servers> progress.getState()
'completed'
ws:/mydomain/serverConfig/Servers>
```

For more information about the `WLSTProgress` object, see "WLSTProgress Object" in `Oracle WebLogic Scripting Tool`.

3.6 Diagnostics Commands

Use the WLST diagnostics commands, listed in Table 3–6, to retrieve diagnostics data
by executing queries against the WebLogic Diagnostics Framework (WLDF) data
stores. For more information about WLDF, see Configuring and Using the Diagnostics Framework for Oracle WebLogic Server.
Table 3–6  Diagnostic Command for WLST Configuration

<table>
<thead>
<tr>
<th>This command...</th>
<th>Enables you to...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>exportDiagnosticData</td>
<td>Execute a query against the specified log file.</td>
<td>Offline</td>
</tr>
<tr>
<td>exportDiagnosticDataFromServer</td>
<td>Executes a query on the server side and retrieves the exported WebLogic Diagnostic Framework (WLDF) data.</td>
<td>Online</td>
</tr>
</tbody>
</table>

3.6.1 exportDiagnosticData

Command Category: Diagnostics Commands
Use with WLST: Offline

3.6.1.1 Description
Executes a query against the specified log file. The results are saved to an XML file.

For more information about the WebLogic Server Diagnostic Service, see Configuring and Using the Diagnostics Framework for Oracle WebLogic Server.

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

3.6.1.2 Syntax
exportDiagnosticData([options])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>options</td>
<td>Optional. Comma-separated list of export diagnostic options, specified as name-value pairs. Valid options include:</td>
</tr>
<tr>
<td></td>
<td>■ beginTimestamp—Timestamp (inclusive) of the earliest record to be added to the result set. This option defaults to 0.</td>
</tr>
<tr>
<td></td>
<td>■ endTimestamp—Timestamp (exclusive) of the latest record to be added to the result set. This option defaults to Long.MAX_VALUE.</td>
</tr>
<tr>
<td></td>
<td>■ exportFileName—Name of the file to which the data is exported. This option defaults to export.xml.</td>
</tr>
<tr>
<td></td>
<td>■ logicalName—Logical name of the log file being read. Valid values include: HarvestedDataArchive, EventsDataArchive, ServerLog, DomainLog, HTTPAccessLog, WebAppLog, ConnectorLog, and JMSMessageLog. This option defaults to ServerLog.</td>
</tr>
<tr>
<td></td>
<td>■ logName—Base log filename containing the log data to be exported. This option defaults to myserver.log.</td>
</tr>
<tr>
<td></td>
<td>■ logRotationDir—Directory containing the rotated log files. This option defaults to &quot;.&quot; (the same directory in which the log file is stored).</td>
</tr>
<tr>
<td></td>
<td>■ query—Expression specifying the filter condition for the data records to be included in the result set. This option defaults to &quot;&quot; (empty string), which returns all data. For more information, see &quot;WLDF Query Language&quot; in Configuring and Using the Diagnostics Framework for Oracle WebLogic Server.</td>
</tr>
<tr>
<td></td>
<td>■ storeDir—Location of the diagnostic store for the server. This option defaults to ../data/store/diagnostics.</td>
</tr>
</tbody>
</table>

3.6.1.3 Example
The following example executes a query against the ServerLog named myserver.log and stores the results in the file named myExport.xml.
3.6.2 exportDiagnosticDataFromServer

Command Category: Diagnostics Commands

Use with WLST: Online

3.6.2.1 Description

Executes a query on the server side and retrieves the exported WebLogic Diagnostic Framework (WLDF) data. The results are saved to an XML file.

For more information about the WebLogic Server Diagnostic Service, see Configuring and Using the Diagnostics Framework for Oracle WebLogic Server.

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

3.6.2.2 Syntax

exportDiagnosticDataFromServer([options])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>options</td>
<td>Optional. Comma-separated list of export diagnostic options, specified as name-value pairs. Valid options include:</td>
</tr>
<tr>
<td></td>
<td>- beginTimestamp—Timestamp (inclusive) of the earliest record to be added to the result set. This option defaults to 0.</td>
</tr>
<tr>
<td></td>
<td>- endTimestamp—Timestamp (exclusive) of the latest record to be added to the result set. This option defaults to Long.MAX_VALUE.</td>
</tr>
<tr>
<td></td>
<td>- exportFileName—Name of the file to which the data is exported. This option defaults to export.xml.</td>
</tr>
<tr>
<td></td>
<td>- logicalName—Logical name of the log file being read. Valid values include: HarvestedDataArchive, EventsDataArchive, ServerLog, DomainLog, HTTPAccessLog, WebAppLog, ConnectorLog, and JMSMessageLog. This option defaults to ServerLog.</td>
</tr>
<tr>
<td></td>
<td>- query—Expression specifying the filter condition for the data records to be included in the result set. This option defaults to &quot;&quot; (empty string), which returns all data.</td>
</tr>
</tbody>
</table>

3.6.2.3 Example

The following example executes a query against the HTTPAccessLog and stores the results in the file named myExport.xml.

wls:/offline/mydomain>exportDiagnosticData(logicalName='ServerLog', logName='myserver.log', exportFileName='myExport.xml')

{'elfFields': '', 'logName': 'myserver.log', 'logRotationDir': '.', 'endTimestamp': 9223372036854775807L, 'exportFileName': 'export.xml', 'storeDir': '../data/store/diagnostics', 'logicalName': 'ServerLog', 'query': '', 'beginTimestamp': 0}

Exporting diagnostic data to export.xml

wls:/offline/mydomain>
3.7 Editing Commands

Use the WLST editing commands, listed in Table 3–7, to interrogate and edit configuration beans.

**Note:** To edit configuration beans, you must be connected to an Administration Server, and you must navigate to the edit tree and start an edit session, as described in Section 3.11.5, “edit” and Section 3.7.17, “startEdit”, respectively.

If you connect to a Managed Server, WLST functionality is limited to browsing the configuration bean hierarchy. While you cannot use WLST to change the values of MBeans on Managed Servers, it is possible to use the Management APIs to do so. Oracle recommends that you change only the values of configuration MBeans on the Administration Server. Changing the values of MBeans on Managed Servers can lead to an inconsistent domain configuration.

For more information about editing configuration beans, see "Using WLST Online to Update an Existing Domain" in *Oracle WebLogic Scripting Tool*.

---

### Table 3–7 Editing Commands for WLST Configuration

<table>
<thead>
<tr>
<th>This command...</th>
<th>Enables you to...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>activate</td>
<td>Activate changes saved during the current editing session but not yet deployed.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td>assign</td>
<td>Assign resources to one or more destinations.</td>
<td>Offline</td>
</tr>
<tr>
<td>cancelEdit</td>
<td>Cancel an edit session, release the edit lock, and discard all unsaved changes. This operation can be called by any user with administrator privileges, even if the user did not start the edit session.</td>
<td>Online</td>
</tr>
<tr>
<td>create</td>
<td>Create a configuration bean of the specified type for the current bean.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td>delete</td>
<td>Delete an instance of a configuration for the current configuration bean.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td>encrypt</td>
<td>Encrypt the specified string.</td>
<td>Online</td>
</tr>
<tr>
<td>get</td>
<td>Return the value of the specified attribute.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td>getActivationTask</td>
<td>Return the latest ActivationTask MBean on which a user can get status.</td>
<td>Online</td>
</tr>
<tr>
<td>invoke</td>
<td>Invokes a management operation on the current configuration bean.</td>
<td>Online</td>
</tr>
<tr>
<td>isRestartRequired</td>
<td>Determine whether a server restart is required.</td>
<td>Online</td>
</tr>
<tr>
<td>loadDB</td>
<td>Load SQL files into a database.</td>
<td>Offline</td>
</tr>
<tr>
<td>loadProperties</td>
<td>Load property values from a file.</td>
<td>Online or Offline</td>
</tr>
</tbody>
</table>
3.7.1 activate

Command Category: Editing Commands

Use with WLST: Online

3.7.1.1 Description

Activates changes saved during the current editing session but not yet deployed. This command prints a message if a server restart is required for the changes that are being activated.

The activate command returns the latest ActivationTask MBean which reflects the state of changes that a user is currently making or has made recently. You can then invoke methods to get information about the latest Configuration Manager activate task in progress or just completed. In the event of an error, the command returns a WLSTException.

3.7.1.2 Syntax

activate([timeout], [block])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>timeout</td>
<td>Optional. Time (in milliseconds) that WLST waits for the activation of configuration changes to complete before canceling the operation. A value of -1 indicates that the operation will not time out. This argument defaults to 300,000 ms (or 5 minutes).</td>
</tr>
</tbody>
</table>
3.7.1.3 Example
The following example activates the changes made during the current edit session that have been saved to disk, but that have not yet been activated. WLST waits for 100,000 ms for the activation to complete, and 200,000 ms before the activation is stopped.

```
wls:/mydomain/edit !> activate(200000, block='true')
Activating all your changes, this may take a while ...
The edit lock associated with this edit session is released once the activation is completed.
Action completed.
wls:/mydomain/edit>
```

3.7.2 assign

Command Category: Editing Commands
Use with WLST: Offline

3.7.2.1 Description
Assigns resources to one or more destinations.
In the event of an error, the command returns a WLSTException.

3.7.2.2 Syntax
assign(sourceType, sourceName, destinationType, destinationName)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>sourceType</td>
<td>Type of configuration bean to be assigned. This value can be set to one of the following values:</td>
</tr>
<tr>
<td></td>
<td>■ AppDeployment</td>
</tr>
<tr>
<td></td>
<td>■ Library</td>
</tr>
<tr>
<td></td>
<td>■ securityType (such as User)</td>
</tr>
<tr>
<td></td>
<td>■ Server</td>
</tr>
<tr>
<td></td>
<td>■ service (such as JDBCSystemResource)</td>
</tr>
<tr>
<td></td>
<td>■ service.SubDeployment, where service specifies the service type of the SubDeployment (such as JMSSystemResource.SubDeployment); you can also specify nested subdeployments (such as AppDeployment.SubDeployment.SubDeployment)</td>
</tr>
</tbody>
</table>

Guidelines for setting this value are provided below.
Use the following guidelines for setting the `sourceType` and `destinationType`:

- When assigning **application deployments**, set the values as follows:
  - `sourceType`: AppDeployment
  - `destinationType`: Target

- When assigning **libraries**, set the values as follows:
  - `sourceType`: Library
  - `destinationType`: Target

- When assigning **services**, set the values as follows:
  - `sourceType`: Name of the specific server, such as JDBCSystemResource
  - `destinationType`: Target

- When assigning **servers to clusters**, set the values as follows:
  - `sourceType`: Server
  - `destinationType`: Cluster

- When assigning **subdeployments**, set the values as follows:
  - `sourceType`: service.SubDeployment, where service specifies the parent of the SubDeployment, such as JMSSystemResource.SubDeployment; you can also specify nested subdeployments (such as AppDeployment.SubDeployment.SubDeployment)
  - `destinationType`: Target

- When assigning **security types**, set the values as follows:
  - `sourceType`: Name of the security type, such as User
  - `destinationType`: Name of the destination security type, such as Group

### 3.7.2.3 Example

The following examples:

- Assign the servers `myServer` and `myServer2` to the cluster `myCluster`. 
Assign all servers to the cluster myCluster.

Assign the application deployment myAppDeployment to the target server newServer.

Assign the user newUser to the group Monitors.

Assign the SubDeployment myQueueSubDeployment, which is a child of the JMS resource myJMSResource, to the target server newServer.

Assign the nested SubDeployment MedRecAppScopedJMS.MedRecJMSServer, which is a child of the AppDeployment AppDeployment, to the target server AdminServer.

### 3.7.3 cancelEdit

Command Category: Editing Commands

Use with WLST: Online

#### 3.7.3.1 Description
Cancels an edit session, releases the edit lock, and discards all unsaved changes.

The user issuing this command does not have to be the current editor; this allows an administrator to cancel an edit session, if necessary, to enable other users to start an edit session.

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

#### 3.7.3.2 Syntax

cancelEdit([defaultAnswer])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>defaultAnswer</td>
<td>Optional. Default response, if you would prefer not to be prompted at the command line. Valid values are y and n. This argument defaults to null, and WLST prompts you for a response.</td>
</tr>
</tbody>
</table>

#### 3.7.3.3 Example
The following example cancels the current editing session. WLST prompts for verification before canceling.

wls:/mydomain/edit !> cancelEdit()
Sure you would like to cancel the edit session? (y/n) y
3.7.4 create

Command Category: Editing Commands

Use with WLST: Online or Offline

3.7.4.1 Description
Creates a configuration bean of the specified type for the current bean. The `create` command returns a stub for the newly created configuration bean. In the event of an error, the command returns a `WLSTException`.

---

**Note:** Child types must be created under an instance of their parent type. You can only create configuration beans that are children of the current Configuration Management Object (`cmo`) type. For more information about the `cmo` variable, see "Changing the Current Management Object" in *Oracle WebLogic Scripting Tool*.

---

Please note the following when using the `create` command with **WLST online**:

- You must be connected to an Administration Server. You cannot use the `create` command for runtime MBeans or when WLST is connected to a Managed Server instance.
- You must navigate to the edit configuration MBean hierarchy using the `edit` command before issuing this command. See Section 3.11.5, "edit".
- You can use the create command to create a WebLogic Server configuration MBean that is a child of the current MBean type.

Please note the following when using the `create` command with **WLST offline**:

- When using WLST offline, the following characters are not valid in object names: period (.), forward slash (/), or backward slash (\).

For more information about:

- Creating MBeans, see "Understanding WebLogic Server MBeans" in *Developing Custom Management Utilities with JMX*.
- Examples of creating specific types of MBean resources, for example, a JMS or JDBC system resource, refer to the WLST sample scripts installed with your product, as described in "WLST Sample Scripts" in *Oracle WebLogic Scripting Tool*.
- MBeans, their child types, attributes, and operations, see *Oracle WebLogic Server MBean Reference*.

3.7.4.2 Syntax

```create(name, childMBeanType, [baseProviderType])```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the configuration bean that you are creating.</td>
</tr>
</tbody>
</table>
3.7.4.3 Example

The following example creates a child configuration bean of type `Server` named `newServer` for the current configuration bean, storing the stub as `server1`:

```
server1=create('newServer','Server')
Server with name 'newServer' has been created successfully.
server1.getName() 'newServer'
```

The following example creates an authentication provider security provider called `myProvider`:

```
cd('SecurityConfiguration/mydomain/Realms/myrealm')
create('myProvider','weblogic.security.providers.authentication.SQLAuthenticator','AuthenticationProvider')
cd('AuthenticationProviders/myProvider')
set('ControlFlag', 'REQUIRED')
```

The following example creates a machine named `highsec_nm` and sets attributes for the associated Node Manager.

```
create('highsec_nm', 'Machine')
cd('Machine/highsec_nm/NodeManager/highsec_nm')
set('DebugEnabled', 'true')
set('ListenAddress', 'innes')
set('NMType', 'SSL')
set('ShellCommand', '')
```

### 3.7.5 delete

Command Category: Editing Commands
Use with WLST: Online or Offline

#### 3.7.5.1 Description

Deletes an instance of a configuration bean of the specified type for the current configuration bean.

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

---

**Note:** You can only delete configuration beans that are children of current Configuration Management Object (`cmo`) type. For more information about the `cmo` variable, see "Changing the Current Management Object" in Oracle WebLogic Scripting Tool.
3.7.5.2 Syntax

```
delete(name, childMBeanType)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the child configuration bean to delete.</td>
</tr>
<tr>
<td>childMBeanType</td>
<td>Type of the configuration bean to be deleted. You can delete instances of any type defined in the config.xml file. For more information about valid configuration beans, see Oracle WebLogic Server MBean Reference.</td>
</tr>
</tbody>
</table>

3.7.5.3 Example

The following example deletes the configuration bean of type `Server` named `newServer`:

```
wls:/mydomain/edit !> delete('newServer','Server')
Server with name 'newServer' has been deleted successfully.
wls:/mydomain/edit !>
```

3.7.6 encrypt

Command Category: Editing Commands

Use with WLST: Online

3.7.6.1 Description

Encrypts the specified string. You can then use the encrypted string in your configuration file or as an argument to a command.

You must invoke this command once for each WebLogic domain in which you want to use the encrypted string. The string can be used only in the WebLogic domain for which it was originally encrypted.

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

3.7.6.2 Syntax

```
encrypt(obj, [domainDir])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>obj</td>
<td>String that you want to encrypt.</td>
</tr>
<tr>
<td>domainDir</td>
<td>Optional. Absolute path name of a WebLogic domain directory. The encrypted string can be used only by the WebLogic domain that is contained within the specified directory. If you do not specify this argument, the command encrypts the string for use in the WebLogic domain to which WLST is currently connected.</td>
</tr>
</tbody>
</table>

3.7.6.3 Example

The following example encrypts the specified string using the `security/SerializedSystemIni.dat` file in the specified WebLogic domain directory.

```
wls:/mydomain/serverConfig>
es=encrypt('myPassword','c:/Oracle/Middleware/domains/mydomain')
```
3.7.7 get

Command Category: Editing Commands
Use with WLST: Online or Offline

3.7.7.1 Description
Returns the value of the specified attribute. For more information about the MBean attributes that can be viewed, see Oracle WebLogic Server MBean Reference. In the event of an error, the command returns a WLSTException.

```
Note: You can list all attributes and their current values by entering ls('a'). For more information, see Section 3.8.12, "ls".
```

Alternatively, you can use the cmo variable to perform any get method on the current configuration bean. For example:

```
cmo.getListenPort()
```

For more information about the cmo variable, see "Changing the Current Management Object" in Oracle WebLogic Scripting Tool.

3.7.7.2 Syntax

```
get(attrName)
```

3.7.7.3 Example

The following example returns the value of the AdministrationPort for the current configuration bean.

```
wls:/mydomain/serverConfig> get('AdministrationPort')
9002
```

Alternatively, you can use the cmo variable:

```
cmo.getAdministrationPort()
```

3.7.8 getActivationTask

Command Category: Editing Commands
Use with WLST: Online

3.7.8.1 Description
Return the latest ActivationTask MBean on which a user can get status. The ActivationTask MBean reflects the state of changes that a user has made recently in WLST. You can then invoke methods to get information about the latest Configuration Manager activate task in progress or just completed. In the event of an error, the command returns a WLSTException.
3.7.8.2 Syntax
getActivationTask()

3.7.8.3 Example
The following example returns the latest ActivationTask MBean on which a user can get status and stores it within the task variable.

wls:/mydomain/edit> task=getActivationTask()
wls:/mydomain/edit> if task!=None:
...  task.getState()
...

3.7.9 invoke

Command Category: Editing Commands
Use with WLST: Online

3.7.9.1 Description
Invokes a management operation on the current configuration bean. Typically, you use this command to invoke operations other than the get and set operations that most WebLogic Server configuration beans provide. The class objects are loaded through the same class loader that is used for loading the configuration bean on which the action is invoked.

You cannot use the invoke command when WLST is connected to a Managed Server instance.

If successful, the invoke command returns the object that is returned by the operation invoked. In the event of an error, the command returns a WLSTException.

3.7.9.2 Syntax
invoke(methodName, parameters, signatures)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>methodName</td>
<td>Name of the method to be invoked.</td>
</tr>
<tr>
<td>parameters</td>
<td>An array of parameters to be passed to the method call.</td>
</tr>
<tr>
<td>signatures</td>
<td>An array containing the signature of the action.</td>
</tr>
</tbody>
</table>

3.7.9.3 Example
The following example invokes the lookupServer method on the current configuration bean.

wls:/mydomain/config> objs = jarray.array([java.lang.String("oamserver")],java.lang.Object)
wls:/mydomain/edit> strs = jarray.array(["java.lang.String"],java.lang.String)
wls:/mydomain/edit> invoke('lookupServer',objs,strs)
true
wls:/mydomain/edit>
3.7.10 isRestartRequired
Command Category: Editing Commands
Use with WLST: Online

3.7.10.1 Description
Determines whether a server restart is required.
If you invoke this command while an edit session is in progress, the response is based on the edits that are currently in progress. If you specify the name of an attribute, WLST indicates whether a server restart is required for that attribute only.
In the event of an error, the command returns a WLSTException.

3.7.10.2 Syntax
isRestartRequired([attributeName])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributeName</td>
<td>Optional. Name of a specific attribute for which you want to check if a server restart is required.</td>
</tr>
</tbody>
</table>

3.7.10.3 Example
The following example specifies whether a server restart is required for all changes made during the current WLST session.

```
[mydomain/edit] > isRestartRequired()
Server re-start is REQUIRED for the set of changes in progress.
```

The following attribute(s) have been changed on MBeans that require server re-start.

```
MBean Changed : mydomain:Name=mydomain,Type=Domain
Attributes changed : AutoConfigurationSaveEnabled
```

The following example specifies whether a server restart is required if you edit the ConsoleEnabled attribute.

```
[mydomain/edit] > isRestartRequired("ConsoleEnabled")
Server re-start is REQUIRED if you change the attribute ConsoleEnabled
```

3.7.11 loadDB
Command Category: Editing Commands
Use with WLST: Offline

3.7.11.1 Description
Loads SQL files into a database.
The loadDB command loads the SQL files from a template file. This command can only be issued after a domain template or extension template has been loaded into memory (see Section 3.3.8, "readDomain" and Section 3.3.9, "readTemplate").
Before executing this command, ensure that the following conditions are true:

- The appropriate database is running.
- SQL files exist for the specified database and version.
To verify that the appropriate SQL files exist, open the domain template and locate the relevant SQL file list, jdbc.index, in the _jdbc_ directory. For example, for PointBase version 4.4, the SQL file list is located at _jdbc_ \Pointbase\44\jdbc.index.

The command fails if the above conditions are not met.

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

### 3.7.11.2 Syntax

loadDB(dbVersion, datasourceName, dbCategory)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>dbVersion</td>
<td>Version of the database for which the SQL files are intended to be used.</td>
</tr>
<tr>
<td>datasourceName</td>
<td>Name of the JDBC data source to be used to load SQL files.</td>
</tr>
<tr>
<td>dbCategory</td>
<td>Optional. Database category associated with the specified data source.</td>
</tr>
</tbody>
</table>

For more information about the jdbc.index file and database categories, see "Files Typically Included in a Template" in the Oracle WebLogic Server Domain Template Reference.

### 3.7.11.3 Example

The following example loads SQL files related to Drop/Create P13N Database Objects intended for version 5.1 of the database, using the p13nDataSource JDBC data source.

wls:/offline/mydomain> loadDB('5.1', 'p13nDataSource', 'Drop/Create P13N Database Objects')

### 3.7.12 loadProperties

Command Category: Editing Commands

Use with WLST: Online and Offline

#### 3.7.12.1 Description

Loads property values from a file and makes them available in the WLST session.

This command cannot be used when you are importing WLST as a Jython module, as described in "Importing WLST as a Jython Module" in Oracle WebLogic Scripting Tool.

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

#### 3.7.12.2 Syntax

loadProperties(fileName)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileName</td>
<td>Properties file pathname.</td>
</tr>
</tbody>
</table>

#### 3.7.12.3 Example

This example gets and sets the properties file values.

wls:/mydomain/serverConfig> loadProperties('c:/temp/myLoad.properties')
3.7.13 save

Command Category: Editing Commands

Use with WLST: Online

3.7.13.1 Description

Saves the edits that have been made but have not yet been saved. This command is only valid when an edit session is in progress. For information about starting an edit session, see Section 3.7.17, "startEdit".

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

3.7.13.2 Syntax

save()

3.7.13.3 Example

The following example saves the edits that have not yet been saved to disk.

wls:/mydomain/edit !> save()
Saving all your changes ...
Saved all your changes successfully.
wls:/mydomain/edit !>

3.7.14 set

Command Category: Editing Commands

Use with WLST: Online or Offline

3.7.14.1 Description

Sets the value of a specified attribute in the current management object. When using WLST offline, this command writes the attribute value to the domain configuration files. When using WLST online, this command sets the value of an MBean attribute. Online changes are written to the domain configuration file when you activate your edits.

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

For information about setting encrypted attributes (all encrypted attributes have names that end with Encrypted), see "Writing and Reading Encrypted Configuration Values" in Oracle WebLogic Scripting Tool.

Note the following when using WLST online:

- You must be in an edit session to use this command. See Section 3.7.17, "startEdit".
- You cannot use this command when WLST is connected to a Managed Server.
- As an alternative to this command, you can use the cmo variable with the following syntax:

  cmo.set attrName (value)

  For example, instead of using set('ListenPort', 7011), you can use:
  cmo.setListenPort(7011)

  For more information about the cmo variable, see "Changing the Current Management Object" in Oracle WebLogic Scripting Tool.
3.7.14.2 Syntax

\[
\text{set(attrName, value)}
\]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>attrName</td>
<td>Name of the attribute to be set.</td>
</tr>
<tr>
<td>value</td>
<td>Value of the attribute to be set.</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>This value should <em>not</em> be enclosed in single or double quotes. See the examples.</td>
</tr>
</tbody>
</table>

3.7.14.3 Example

The following example sets the ArchiveConfigurationCount attribute of DomainMBean to 10:

\[\text{wls:/mydomain/serverConfig} \text{ set('ArchiveConfigurationCount', 10)}\]

The following example sets the long value of the T1TimerInterval attribute of a custom Mbean to 123:

\[\text{wls:/mydomain/serverConfig} \text{ set('T1TimerInterval', Long(123))}\]

The following example sets the boolean value of the MyBooleanAttribute attribute of a custom Mbean to true:

\[\text{wls:/mydomain/serverConfig} \text{ set('MyBooleanAttribute', Boolean(true))}\]

3.7.15 setOption

Command Category: Editing Commands

Use with WLST: Offline

3.7.15.1 Description

Sets options related to a WebLogic domain creation or update. In the event of an error, the command returns a WLSTException.

3.7.15.2 Syntax

\[
\text{setOption(optionName, optionValue)}
\]
<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>optionName</code></td>
<td>Name of the option to set.</td>
</tr>
</tbody>
</table>

Available options for **domain creation** include:

- **CreateStartMenu**—Boolean value specifying whether to create a Start Menu shortcut on a Windows platform. This option defaults to `true`.
  
  **Note:** If a user with Administrator privileges installed the software and chose to create the Start menu entries in the All Users folder, only users with Administrator privileges can create Start menu entries in the same folder when creating a WebLogic domain using the Configuration Wizard or WLST. That is, if a user without Administrator privileges uses the Configuration Wizard or WLST from this installation to create domains, Start menu shortcuts to the domains are not created. In this case, the users can manually create shortcuts in their local Start menu folder, if desired.

- **DomainName**—Name of the WebLogic domain. By default, the name of the WebLogic domain is derived from the name of the domain directory. For example, for a WebLogic domain saved to `c:/Oracle/Middleware/user_projects/domains/myMedrec`, the domain name is `myMedrec`. By setting `DomainName`, the name of the created domain will be independent of the domain directory name.

- **JavaHome**—Home directory for the JVM to be used when starting the server. The default for this option depends on the platform on which you install WebLogic Server.

- **OverwriteDomain**—Boolean value specifying whether to allow an existing WebLogic domain to be overwritten. This option defaults to `false`.

- **ServerStartMode**—Mode to use when starting the server for the newly created WebLogic domain. This value can be `dev` (development) or `prod` (production). This option defaults to `dev`.

Available options for **domain updates** include:

- **AllowCasualUpdate**—Boolean value specifying whether to allow a WebLogic domain to be updated without adding an extension template. This option defaults to `true`.

- **ReplaceDuplicates**—Boolean value specifying whether to keep original configuration elements in the WebLogic domain or replace the elements with corresponding ones from an extension template when there is a conflict. This option defaults to `true`.

Available options for both **domain creation** and **domain updates** include:

- **AppDir**—Application directory to be used when a separate directory is desired for applications, as specified by the template. This option defaults to `WL_HOME/user_projects/applications/domainname`, where `WL_HOME` specifies the WebLogic Server home directory and `domainname` specifies the name of the WebLogic domain.

- **AutoAdjustSubDeploymentTarget**—Boolean value specifying whether WLST automatically adjusts targets for the subdeployments of AppDeployments. This option defaults to `true`. To deactivate this feature, set the option to `false` and explicitly set the targeting for AppDeployment subdeployments before writing or updating the WebLogic domain or domain template.

- **AutoDeploy**—Boolean value specifying whether to activate auto deployment when a cluster or multiple Managed Servers are created. This option defaults to `true`. To deactivate this feature, set the option to `false` on the first line of your script.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>optionValue</code></td>
<td>Value for the option.</td>
</tr>
</tbody>
</table>

**Note:** Boolean values can be specified as a String (`true`, `false`) or integer (0, 1).
3.7.15.3 Example
The following example sets the CreateStartMenu option to false:

```
wlsh:/offline> setOption('CreateStartMenu', 'false')
```

3.7.16 showChanges

Command Category: Editing Commands
Use with WLST: Online

3.7.16.1 Description
Shows the changes made to the configuration by the current user during the current edit session. In the event of an error, the command returns a WLSTException.

3.7.16.2 Syntax
```
showChanges([onlyInMemory])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>onlyInMemory</td>
<td>Optional. Boolean value specifying whether to display only the changes that have not yet been saved. This argument defaults to false, indicating that all changes that have been made from the start of the session are displayed.</td>
</tr>
</tbody>
</table>

3.7.16.3 Example
The following example shows all of the changes made by the current user to the configuration since the start of the current edit session.

```
wls:/mydomain/edit !> showChanges()
```
Changes that are in memory and saved to disc but not yet activated are:

```
MBean Changed           : com.bea:Name=basicWLSDomain,Type=Domain
Operation Invoked       : add
Attribute Modified      : Machines
Attributes Old Value    : null
Attributes New Value    : Mach1
Server Restart Required : false

MBean Changed           : com.bea:Name=basicWLSDomain,Type=Domain
Operation Invoked       : add
Attribute Modified      : Servers
Attributes Old Value    : null
Attributes New Value    : myserver
Server Restart Required : false
```

3.7.17 startEdit

Command Category: Editing Commands
Use with WLST: Online

3.7.17.1 Description
Starts a configuration edit session on behalf of the currently connected user. You must navigate to the edit configuration MBean hierarchy using the edit command before issuing this command. For more information, see Section 3.11.5, "edit".
This command must be called prior to invoking any command to modify the WebLogic domain configuration.

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

**Note:** WLST automatically starts an edit session if it detects that there is an edit session that is already in progress by the same user, which may have been started via the Administration Console or another WLST session.

### 3.7.17.2 Syntax

```
startEdit([waitTimeInMillis], [timeoutInMillis], [exclusive])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>waitTimeInMillis</td>
<td>Optional. Time (in milliseconds) that WLST waits until it gets a lock, in the event that another user has a lock. This argument defaults to 0 ms.</td>
</tr>
<tr>
<td>timeoutInMillis</td>
<td>Optional. Timeout (in milliseconds) that WLST waits to release the edit lock. This argument defaults to -1 ms, indicating that this edit session never expires.</td>
</tr>
<tr>
<td>exclusive</td>
<td>Optional. Specifies whether the edit session should be an exclusive session. If set to true, if the same owner enters the startEdit command, WLST waits until the current edit session lock is released before starting the new edit session. The exclusive lock times out according to the time specified in timeoutInMillis. This argument defaults to false.</td>
</tr>
</tbody>
</table>

### 3.7.17.3 Example

The following example saves the edits that have not yet been saved to disk.

```
wls:/mydomain/edit> startEdit(60000, 120000)
Starting an edit session ...
Started edit session, please be sure to save and activate your changes once you are done.
```

### 3.7.18 stopEdit

**Command Category:** Editing Commands

**Use with WLST:** Online

#### 3.7.18.1 Description

Stops the current edit session, releases the edit lock, and discards unsaved changes.

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

#### 3.7.18.2 Syntax

```
stopEdit([defaultAnswer])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>defaultAnswer</td>
<td>Optional. Default response, if you would prefer not to be prompted at the command line. Valid values are y and n. This argument defaults to null, and WLST prompts you for a response.</td>
</tr>
</tbody>
</table>
3.7.18.3 Example
The following example stops the current editing session. WLST prompts for verification before canceling.

```
wls:/mydomain/edit !> stopEdit()
Sure you would like to stop your edit session? (y/n)
y
Edit session has been stopped successfully.
wls:/mydomain/edit>
```

3.7.19 unassign

Command Category: Editing Commands

Use with WLST: Offline

3.7.19.1 Description
Unassign applications or resources from one or more destinations.

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

3.7.19.2 Syntax

```
unassign(sourceType, sourceName, destinationType, destinationName)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>sourceType</td>
<td>Type of configuration bean to be unassigned. This value can be set to one of the following values:</td>
</tr>
<tr>
<td></td>
<td>■ AppDeployment</td>
</tr>
<tr>
<td></td>
<td>■ Library</td>
</tr>
<tr>
<td></td>
<td>■ securityType (such as User)</td>
</tr>
<tr>
<td></td>
<td>■ Server</td>
</tr>
<tr>
<td></td>
<td>■ service (such as JDBCSystemResource)</td>
</tr>
<tr>
<td></td>
<td>■ service.SubDeployment, where service specifies the service type of the SubDeployment (such as JMSSystemResource.SubDeployment); you can also specify nested subdeployments (such as AppDeployment.SubDeployment.SubDeployment)</td>
</tr>
<tr>
<td>sourceName</td>
<td>Name of the application or resource to be unassigned. Multiple names can be specified, separated by commas, or you can use the wildcard (*) character to specify all resources of the specified type.</td>
</tr>
<tr>
<td></td>
<td>Specify subdeployments using the following format: service.subDeployment, where service specifies the parent service and subDeployment specifies the name of the subdeployment. For example, myJMSResource.myQueueSubDeployment. You can also specify nested subdeployments, such as MedRecEAR.MedRecAppScopedJMS.MedRecJMSServer.</td>
</tr>
<tr>
<td>destinationType</td>
<td>Type of destination. Guidelines for setting this value are provided below.</td>
</tr>
<tr>
<td>destinationName</td>
<td>Name of the destination. Multiple names can be specified, separated by commas.</td>
</tr>
</tbody>
</table>

Use the following guidelines for setting the sourceType and destinationType:

- When unassigning application deployments, set the values as follows:
- `sourceType`: AppDeployment
- `destinationType`: Target

When unassigning **libraries**, set the values as follows:
- `sourceType`: Library
- `destinationType`: Target

When unassigning **security types**, set the values as follows:
- `sourceType`: Name of the security type, such as User
- `destinationType`: Name of the destination security type, such as Group

When unassigning **servers** from clusters, set the values as follows:
- `sourceType`: Server
- `destinationType`: Cluster

When unassigning **services**, set the values as follows:
- `sourceType`: Name of the specific server, such as JDBCSystemResource
- `destinationType`: Target

When unassigning **subdeployments**, set the values as follows:
- `sourceType`: `service.SubDeployment`, where `service` specifies the parent of the SubDeployment, such as `JMSSystemResource.SubDeployment`; you can also specify nested subdeployments (such as `AppDeployment.SubDeployment.SubDeployment`)
- `destinationType`: Target

### 3.7.19.3 Example
The following examples:

- **Unassign the servers** `myServer` and `myServer2` from the cluster `myCluster`.
  ```
  wls:/offline/medrec> unassign("Server", "myServer,myServer2", "Cluster", "myCluster")
  ```

- **Unassign all servers** from the cluster `myCluster`.
  ```
  wls:/offline/mydomain> unassign("Server", ",", "Cluster", "myCluster")
  ```

- **Unassign the user** `newUser` from the group `Monitors`.
  ```
  wls:/offline/medrec> unassign("User", "newUser", "Group", "Monitors")
  ```

- **Unassign the application deployment** `myAppDeployment` from the target server `newServer`.
  ```
  wls:/offline/mydomain> unassign("AppDeployment", "myAppDeployment", "Target", "newServer")
  ```

- **Unassign the nested SubDeployment** `MedRecAppScopedJMS.MedRecJMSServer`, which is a child of the AppDeployment `AppDeployment`, from the target server `AdminServer`.
  ```
  ```
3.7.20 undo

Command Category: Editing Commands
Use with WLST: Online

3.7.20.1 Description
Reverts all unsaved or unactivated edits.
You specify whether to revert all unactivated edits (including those that have been saved to disk), or all edits made since the last save operation. This command does not release the edit session.
In the event of an error, the command returns a WLSTException.

3.7.20.2 Syntax
undo([unactivatedChanges], [defaultAnswer])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>unactivatedChanges</td>
<td>Optional. Boolean value specifying whether to undo all unactivated changes, including edits that have been saved to disk. This argument defaults to false, indicating that all edits since the last save operation are reverted.</td>
</tr>
<tr>
<td>defaultAnswer</td>
<td>Optional. Default response; if you would prefer not to be prompted at the command line. Valid values are y and n. This argument defaults to null, and WLST prompts you for a response.</td>
</tr>
</tbody>
</table>

3.7.20.3 Example
The following example reverts all changes since the last save operation. WLST prompts for verification before reverting.

```bash
wls:/mydomain/edit !> undo()
Sure you would like to undo your changes? (y/n)
y
Discarded your in-memory changes successfully.
wls:/mydomain/edit>
```

The following example reverts all unactivated changes. WLST prompts for verification before reverting.

```bash
wls:/mydomain/edit !> undo('true')
Sure you would like to undo your changes? (y/n)
y
Discarded all your changes successfully.
wls:/mydomain/edit>
```

3.7.21 validate

Command Category: Editing Commands
Use with WLST: Online

3.7.21.1 Description
Validates the changes that have been made but have not yet been saved. This command enables you to verify that all changes are valid before saving them.
In the event of an error, the command returns a WLSTException.
3.7.21.2 Syntax
validate()

3.7.21.3 Example
The following example validates all changes that have been made but have not yet been saved.

wls:/mydomain/edit !> validate()
Validating changes ... 
Validated the changes successfully

3.8 Information Commands
Use the WLST information commands, listed in Table 3–8, to interrogate domains, servers, and variables, and provide configuration bean, runtime bean, and WLST-related information.

Table 3–8 Information Commands for WLST Configuration

<table>
<thead>
<tr>
<th>This command...</th>
<th>Enables you to...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>addListener</td>
<td>Add a JMX listener to the specified MBean.</td>
<td>Online</td>
</tr>
<tr>
<td>configToScript</td>
<td>Convert an existing server configuration (config directory) to an executable WLST script</td>
<td>Online or Offline</td>
</tr>
<tr>
<td>dumpStack</td>
<td>Display stack trace from the last exception that occurred while performing a WLST action, and reset the stack trace.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td>dumpVariables</td>
<td>Display all variables used by WLST, including their name and value.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td>find</td>
<td>Find MBeans and attributes in the current hierarchy.</td>
<td>Online</td>
</tr>
<tr>
<td>getConfigManager</td>
<td>Return the latest ConfigurationManagerBean MBean which manages the change process.</td>
<td>Online</td>
</tr>
<tr>
<td>getMBean</td>
<td>Return the MBean by browsing to the specified path.</td>
<td>Online</td>
</tr>
<tr>
<td>getMBI</td>
<td>Return the MBeanInfo for the specified MBeanType or the cmo variable.</td>
<td>Online</td>
</tr>
<tr>
<td>getPath</td>
<td>Return the MBean path for the specified MBean instance.</td>
<td>Online</td>
</tr>
<tr>
<td>listChildTypes</td>
<td>List all the children MBeans that can be created or deleted for the cmo type.</td>
<td>Online</td>
</tr>
<tr>
<td>lookup</td>
<td>Look up the specified MBean.</td>
<td>Online</td>
</tr>
<tr>
<td>ls</td>
<td>List all child beans and/or attributes for the current configuration or runtime bean.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td>man</td>
<td>Display help from MBeanInfo for the current MBean or its specified attribute.</td>
<td>Online</td>
</tr>
<tr>
<td>redirect</td>
<td>Redirect WLST output to the specified filename.</td>
<td>Online</td>
</tr>
<tr>
<td>removeListener</td>
<td>Remove a listener that was previously defined.</td>
<td>Online</td>
</tr>
</tbody>
</table>
3.8.1 addListener

Command Category: Information Commands

Use with WLST: Online

3.8.1.1 Description

Adds a JMX listener to the specified MBean. Any changes made to the MBean are reported to standard out and/or are saved to the specified configuration file.

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

3.8.1.2 Syntax

addListener(mbean, [attributeNames], [logFile], [listenerName])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>mbean</td>
<td>Name of the MBean or MBean object to listen on.</td>
</tr>
<tr>
<td>attributeName</td>
<td>Optional. Comma-separated list of all attribute names on which you would like to add a JMX listener. This argument defaults to null, and adds a JMX listener for all attributes.</td>
</tr>
<tr>
<td>logFile</td>
<td>Optional. Name and location of the log file to which you want to write listener information. This argument defaults to standard out.</td>
</tr>
<tr>
<td>listenerName</td>
<td>Optional. Name of the JMX listener. This argument defaults to a WLST-generated name.</td>
</tr>
</tbody>
</table>

3.8.1.3 Example

The following example defines a JMX listener on the cmo MBean for the Notes and ArchiveConfigurationCount attributes. The listener is named domain-listener and is stored in ./listeners/domain.log.
3.8.2 configToScript

Command Category: Information Commands
Use with WLST: Online or Offline

Converts an existing server configuration (config directory) to an executable WLST script. You can use the resulting script to re-create the resources on other servers.

The configToScript command creates the following files:

- A WLST script that contains the commands needed to recreate the configuration.
- A properties file that contains domain-specific values. You can update the values in this file to create new domains that are similar to the original configuration.
- A user configuration file and an associated key file to store encrypted attributes. The user configuration file contains the encrypted information. The key file contains a secret key that is used to encrypt and decrypt the encrypted information.

When you run the generated script:

- If a server is currently running, WLST will try to connect using the values in the properties file and then run the script commands to create the server resources.
- If no server is currently running, WLST will start a server with the values in the properties file, run the script commands to create the server resources, and shutdown the server. This may cause WLST to exit from the command shell.

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

3.8.2.1 Syntax

configToScript([configPath], [pyPath], [overwrite], [propertiesFile], [createDeploymentScript])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>configPath</td>
<td>Optional. Path to the domain directory that contains the configuration that you want to convert. This argument defaults to the directory from which you start WLST (./).</td>
</tr>
<tr>
<td>pyPath</td>
<td>Optional. Path and filename to which you want to write the converted WLST script. This argument defaults to ./config/config.py.</td>
</tr>
<tr>
<td>overwrite</td>
<td>Optional. Boolean value specifying whether the script file should be overwritten if it already exists. This argument defaults to true, indicating that the script file is overwritten.</td>
</tr>
<tr>
<td>propertiesFile</td>
<td>Optional. Path to the directory in which you want WLST to write the properties files. This argument defaults to the pathname specified for the scriptPath argument.</td>
</tr>
<tr>
<td>createDeploymentScript</td>
<td>Optional. Boolean value specifying whether WLST creates a script that performs deployments only. This argument defaults to false, indicating that a deployment script is not created.</td>
</tr>
</tbody>
</table>

3.8.2.2 Example

The following example converts the configuration to a WLST script config.py. By default, the configuration file is loaded from ./config, the script file is saved to
.config/config.py, and the properties files is saved to .config/config.py.properties.

`confiToScript()`

`confiToScript` is loading configuration from c:\Oracle\Middleware\user_projects\domains\wls\config\config.xml ... Completed configuration load, now converting resources to wlst script... `confiToScript` completed successfully

The WLST script is written to c:\Oracle\Middleware\user_projects\domains\wls\config\config.py and the properties file associated with this script is written to c:\Oracle\Middleware\user_projects\domains\wls\config\config.py.properties

The following example converts server resources configured in the file c:\Oracle\Middleware\user_projects\domains\mydomain\config directory to a WLST script c:\Oracle\Middleware\myscripts\config.py.

`confiToScript('c:/Oracle/Middleware/user_projects/domains/mydomain','c:/Oracle/Middleware/myscripts')`

`confiToScript` is loading configuration from c:\Oracle\Middleware\user_projects\domains\mydomain\config\config.xml ... Completed configuration load, now converting resources to wlst script... `confiToScript` completed successfully

The WLST script is written to c:\Oracle\Middleware\myscripts\config.py and the properties file associated with this script is written to c:\Oracle\Middleware\mydomain\config.py.properties

**3.8.3 dumpStack**

Command Category: Information Commands

Use with WLST: Online or Offline

**3.8.3.1 Description**

Displays the stack trace from the last exception that occurred while performing a WLST action, and resets the stack trace.

If successful, the `dumpstack` command returns the Throwable object. In the event of an error, the command returns a `WLSTException`.

**3.8.3.2 Syntax**

`dumpStack()`

**3.8.3.3 Example**

This example displays the stack trace.

`wls:/myserver/serverConfig> dumpStack()`

`com.bea.plateng.domain.script.jython.WLSTException: java.lang.reflect.InvocationTargetException` `java.lang.reflect.InvocationTargetException` `TargetException` ...

**3.8.4 dumpVariables**

Command Category: Information Commands

Use with WLST: Online or Offline
3.8.4.1 **Description**  
Displays all the variables used by WLST, including their name and value. In the event of an error, the command returns a WLSTException.

3.8.4.2 **Syntax**  
dumpVariables()

3.8.4.3 **Example**  
This example displays all the current variables and their values.

```
wls:/mydomain/serverConfig> dumpVariables()
adminHome   weblogic.rmi.internal.BasicRemoteRef - hostID: '-1 10808015094263937S:localhost:[7001,8001,-1,-1,-1,-1,-1]:
myDomain:AdminServer', oid: '259', channel: 'null'
cmgr   [MBeanServerInvocationHandler]com.bea:Name=ConfigurationManager,
  Type=weblogic.management.mbeanservers.edit.ConfigurationManagerMBean
cmo   [MBeanServerInvocationHandler]com.bea:Name=mydomain,Type=Domain
connected true
domainName mydomain
...
wls:/mydomain/serverConfig>
```

3.8.5 **find**  
Command Category: Information Commands  
Use with WLST: Online

3.8.5.1 **Description**  
Finds MBeans and attributes in the current hierarchy.  
WLST returns the pathname to the MBean that stores the attribute and/or attribute type, and its value. If searchInstancesOnly is set to false, this command also searches the MBeanType paths that are not instantiated in the server, but that can be created. In the event of an error, the command returns a WLSTException.

3.8.5.2 **Syntax**  
find([name], [type], [searchInstancesOnly])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Optional. Name of the attribute to find.</td>
</tr>
<tr>
<td>type</td>
<td>Optional. Type of the attribute to find.</td>
</tr>
<tr>
<td>searchInstancesOnly</td>
<td>Optional. Boolean value specifying whether to search registered instances only or to also search MBeanTypes paths that are not instantiated in the server, but that can be created. This argument defaults to true, indicating only the registered instances will be searched.</td>
</tr>
</tbody>
</table>

3.8.5.3 **Example**  
The following example searches for an attribute named `javaCompiler` in the current configuration hierarchy.

```
wls:/mydomain/serverConfig> find(name = 'JavaCompiler')
Finding 'JavaCompiler' in all registered MBean instances ...
```
The following example searches for an attribute of type JMSRuntime in the current configuration hierarchy.

```
wlst:/mydomain/serverRuntime> find(type='JMSRuntime')
Finding MBean of type 'JMSRuntime' in all the instances ...
/JMSRuntime/AdminServer.jms
wlst:/mydomain/serverRuntime>
```

The following example searches for an attribute named execute in the current configuration hierarchy. The searchInstancesOnly argument is set to false, indicating to also search MBeanTypes that are not instantiated in the server.

```
wls:/mydomain/serverConfig> find(name='execute', searchInstancesOnly='false')
Finding 'execute' in all registered MBean instances ...
/Servers/AdminServer      ExecuteQueues [Ljava.lang.Object;@1aa7dbc
/Servers/AdminServer       Use81StyleExecuteQueues                            false
Now finding 'execute' in all MBean Types that can be instantiated ...
/Servers                                      ExecuteQueues
/Servers                                      Use81StyleExecuteQueues
wls:/mydomain/serverConfig>
```

### 3.8.6 getConfigManager

**Command Category:** Information Commands

**Use with WLST:** Online

**3.8.6.1 Description**

Returns the latest ConfigurationManager MBean which manages the change process. You can then invoke methods to manage configuration changes across a WebLogic domain. In the event of an error, the command returns a WLSTException.

**3.8.6.2 Syntax**

```
getConfigManager()
```

**3.8.6.3 Example**

The following example returns the latest ConfigurationManagerBean MBean and stores it within the task variable.

```
wls:/mydomain/serverConfig> cm=getConfigManager()
wls:/mydomain/serverConfig> cm.getType() 'weblogic.management.mbeanservers.edit.ConfigurationManagerMBean'
```

### 3.8.7 getMBean

**Command Category:** Information Commands

**Use with WLST:** Online

**3.8.7.1 Description**

Returns the MBean by browsing to the specified path. In the event of an error, the command returns a WLSTException.
3.8.7.2 Syntax
getMBean(mbeanPath)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>mbeanPath</td>
<td>Path name to the MBean in the current hierarchy.</td>
</tr>
</tbody>
</table>

3.8.7.3 Example
The following example returns the MBean specified by the path.

wls:/mydomain/edit !> com=getMBean('Servers/myserver/COM/myserver')
wls:/mydomain/edit !> com.getType()
'Server'

3.8.8 getMBI

Command Category: Information Commands
Use with WLST: Online

3.8.8.1 Description
Returns the MBeanInfo for the specified MBeanType or the cmo variable. In the event of an error, the command returns a WLSTException.

3.8.8.2 Syntax
getMBI([mbeanType])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>mbeanType</td>
<td>Optional. MBeanType for which the MBeanInfo is displayed.</td>
</tr>
</tbody>
</table>

3.8.8.3 Example
The following example gets the MBeanInfo for the specified MBeanType and stores it in the variable svrMbi.

wls:/mydomain/serverConfig>
svrMbi=getMBI('weblogic.management.configuration.ServerMBean')

3.8.9 getPath

Command Category: Information Commands
Use with WLST: Online

3.8.9.1 Description
Returns the MBean path for the specified MBean instance or ObjectName for the MBean in the current tree. In the event of an error, the command returns a WLSTException.

3.8.9.2 Syntax
getPath(mbean)
3.8.9.3 Example
The following example returns the MBean specified by the path.

```
wlst:/mydomain/edit !> path=getPath('com.bea:Name=myserver,Type=Server')
wlst:/mydomain/edit !> print path
'Servers/myserver'
```

3.8.10 listChildTypes
Command Category: Information Commands
Use with WLST: Online

3.8.10.1 Description
Lists all the child MBeans that can be created or deleted for the `cmo`. The `cmo` variable specifies the configuration bean instance to which you last navigated using WLST. For more information about the `cmo` variable, see "Changing the Current Management Object" in Oracle WebLogic Scripting Tool.

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

3.8.10.2 Syntax
```
listChildTypes([parent])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>parent</td>
<td>Optional. Parent type for which you want the children types listed.</td>
</tr>
</tbody>
</table>

3.8.10.3 Example
The following example lists the children MBeans that can be created or deleted for the `cmo` type.

```
wlst:/mydomain/serverConfig> listChildTypes()
AppDeployments
BridgeDestinations
CachingRealms
Clusters
...
wlst:/mydomain/serverConfig>
```

3.8.11 lookup
Command Category: Information Commands
Use with WLST: Online

3.8.11.1 Description
Looks up the specified MBean. The MBean must be a child of the current MBean. In the event of an error, the command returns a `WLSTException`. 
### 3.8.11.2 Syntax

```
lookup(name, [childMBeanType])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the MBean that you want to lookup.</td>
</tr>
<tr>
<td>childMBeanType</td>
<td>Optional. The type of the MBean that you want to lookup.</td>
</tr>
</tbody>
</table>

### 3.8.11.3 Example

The following example looks up the specified server, `myserver`, and stores the returned stub in the `sbean` variable.

````
wls:/mydomain/serverConfig> sbean=lookup('myserver','Server')
wls:/mydomain/serverConfig> sbean.getType()
'Server'
```

### 3.8.12 ls

**Command Category:** Information Commands  
**Use with WLST:** Online or Offline

#### 3.8.12.1 Description

Lists the attributes, operations, and child management objects of the specified management object.

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

By default, the output is returned as a string and is arranged in three columns:

- The first column displays a set of codes that describe the listed item. See Table 3–9.
- The second column displays the item name.
- When the item is an attribute, the third column displays the attribute value. If an attribute is encrypted, the third column displays asterisks instead of the value. (See "Writing and Reading Encrypted Configuration Values" in Oracle WebLogic Scripting Tool.)
- When the item is an operation, the third column uses the following pattern to display the operation's return type and input parameters: `returnType: parameterType(parameterName)`

#### Table 3–9  ls Command Output Information

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
</table>
| d    | Indicates that the item is a child management object.  
Like a directory in a UNIX or Windows file system, you can use the `cd` command to make the child object the current management object. |
| r    | Indicates that the item is a child management object or an attribute that is readable, assuming that current user has been given read permission by the security realm’s policies. (See "Default Security Policies for MBeans” in the Oracle WebLogic Server MBean Reference.) |
| w    | Indicates that the item is an attribute that is writable, assuming that current user has been given write permission by the security realm’s policies. (See "Default Security Policies for MBeans” in the Oracle WebLogic Server MBean Reference.) |
By default, the output lists all attributes, operations, and child management objects of the current management object. To filter the output or to see a list for a different management object, you can specify a command argument.

### Argument Definition

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>Optional. Displays only the attributes of the specified management object (suppresses the display of other items).</td>
</tr>
<tr>
<td>c</td>
<td>Optional. Displays only the child management objects of the specified management object (suppresses the display of other items).</td>
</tr>
<tr>
<td>o</td>
<td>Optional. Displays only the operations that can be invoked on the specified management object (suppresses the display of other items). This argument is only applicable for WLST online.</td>
</tr>
</tbody>
</table>

### Syntax

```
ls( [ a | c | o ] [ moPath ])
```

```
ls( [ moPath ] returnMap [ returnType ] )
```

**Note:** As a performance optimization, when using WLST offline, WebLogic Server does not store most of its default values in the configuration files for the WebLogic domain. In some cases, this optimization prevents entire management objects from being displayed by WLST offline (because WebLogic Server has never written the corresponding XML elements to the domain configuration files). For example, if you never modify the default logging severity level for a WebLogic domain while the domain is active, WLST offline will not display the `Log` management object for the domain.

If you want to change the default value of attributes whose management object is not displayed by WLST offline, you must first use the `create` command to create the management object. Then you can `cd` to the management object and change the attribute value. See Section 3.7.4, "create".
3.8.12.3 Example

The following example displays all the child configuration beans, and attribute names and values for the examples domain, which has been loaded into memory, in WLST offline mode:

```
wlst:/offline/mydomain > ls()
```

```
dr--   AppDeployments
dr--   BridgeDestinations
dr--   Clusters
dr--   CustomResources
dr--   DeploymentConfiguration
dr--   Deployments
dr--   EmbeddedLDAP
dr--   ErrorHandlings
dr--   FileStores
dr--   InternalAppDeployments
dr--   InternalLibraries
dr--   JDBCDataSourceFactories
dr--   JDBCStores
dr--   JDBCSystemResources
dr--   JMSBridgeDestinations
dr--   JMSInteropModules
dr--   JMSServers
dr--   JMSSystemResources
dr--   JMX
...
wlst:/offline/examples>
```

The following example displays all the attribute names and values in DomainMBean:

```
wls:/mydomain/serverConfig> ls('a')
```

```
-r--   AdminServerName                              AdminServer
-r--   AdministrationMBeanAuditingEnabled           false
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>moPath</td>
<td>Optional. Path name to the management object for which you want to list attributes, operations, and child management objects. You can specify a pathname that is relative to your current location in the hierarchy or an absolute pathname. With WLST offline, use the forward-slash character (/) to specify the root of the configuration document. With WLST online, you can list the contents of MBeans in any management hierarchy (see Section 3.11, &quot;Tree Commands&quot;). Use the following syntax to specify the root of a hierarchy: root-name:/ For example, to list the root of the server runtime hierarchy: ls('serverRuntime:/') If you do not specify this argument, the command lists items for the current management object.</td>
</tr>
<tr>
<td>returnType</td>
<td>Optional. Controls the output returned in the map. Specify a, c, or o, which filter the output as described at the top of this table. This argument is valid only if returnMap is set to true. This argument defaults to c.</td>
</tr>
<tr>
<td>returnMap</td>
<td>Optional. Boolean value that determines whether the command returns values as a map. This argument defaults to false, which causes this command to return a String.</td>
</tr>
</tbody>
</table>
The following example displays all the child beans and attribute names and values in Servers MBean:

```
wlst:/mydomain/serverConfig> ls('Servers')
```
dr-- AdminServer

The following example displays the attribute names and values for the specified MBean path and returns the information in a map:

```
wlst:/mydomain/serverConfig> svrAttrList = ls('edit:/Servers/myserver', 'true', 'a')
```

```
-rw- AcceptBacklog                                50
-rw- AdminReconnectIntervalSeconds                10
-rw- AdministrationPort                           9002
-rw- AdministrationProtocol                       t3s
-rw- AutoKillIfFailed                               false
-rw- AutoMigrationEnabled                          false
-rw- AutoRestart                                   true
-rw- COMEnabled                                    false
-rw- ClasspathServletDisabled                      false
-rw- ClientCertProxyEnabled                        false
-rw- Cluster                                       null
-rw- ClusterRuntime                                null
-rw- ClusterWeight                                 100
```

### 3.8.13 man

**Command Category:** Information Commands

**Use with WLST:** Online

#### 3.8.13.1 Description

Displays help from MBeanInfo for the current MBean or its specified attribute. In the event of an error, the command returns a WLSTException.

#### 3.8.13.2 Syntax

```
man([attrName])
```
### 3.8.13.3 Example

The following example displays help from `MBeanInfo` for the `ServerMBean` bean.

```
3.8.13.3 Example
The following example displays help from MBeanInfo for the ServerMBean bean.

wls:/mydomain/serverConfig> man('Servers')
dynamic : true
creator : createServer
destroyer : destroyServer
description : <p>Returns the ServerMBeans representing the servers that have been configured to be part of this domain.</p>
descriptorType : Attribute
Name : Servers
interfaceClassName : [Lweblogic.management.configuration.ServerMBean;
displayName : Servers
relationship : containment
```

### 3.8.14 redirect

Command Category: Information Commands

Use with WLST: Online

#### 3.8.14.1 Description

Redirects WLST information, error, and debug messages to the specified filename. Also redirects the output of the `dumpStack()` and `dumpVariables()` commands to the specified filename.

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

#### 3.8.14.2 Syntax

```
redirect(outputFile, [toStdOut])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>outputFile</td>
<td>Name of the file to which you want to record the WLST commands. The filename can be absolute or relative to the directory from which you started WLST.</td>
</tr>
<tr>
<td>toStdOut</td>
<td>Optional. Boolean value specifying whether the output should be sent to stdout. This argument defaults to true, indicating that the output will be sent to stdout.</td>
</tr>
</tbody>
</table>

#### 3.8.14.3 Example

The following example begins redirecting WLST output to the `logs/wlst.log` file:

```
wls:/mydomain/serverConfig> redirect('./logs/wlst.log')
```

### 3.8.15 removeListener

Command Category: Information Commands

Use with WLST: Online
3.8.15.1 Description
Removes a listener that was previously defined. If you do not specify an argument, WLST removes all listeners defined for all MBeans. For information about setting a listener, see Section 3.8.1, "addListener".
In the event of an error, the command returns a WLSTException.

3.8.15.2 Syntax
removeListener([mbean], [listenerName])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>mbean</td>
<td>Optional. Name of the MBean or MBean object for which you want to remove the previously defined listeners.</td>
</tr>
<tr>
<td>listenerName</td>
<td>Optional. Name of the listener to be removed.</td>
</tr>
</tbody>
</table>

3.8.15.3 Example
The following example removes the listener named mylistener.
wls:/mydomain/serverConfig> removeListener(listenerName="mylistener")

3.8.16 showListeners
Command Category: Information Commands
Use with WLST: Online

3.8.16.1 Description
Shows all listeners that are currently defined. For information about setting a listener, see Section 3.8.1, "addListener".
In the event of an error, the command returns a WLSTException.

3.8.16.2 Syntax
showListeners()

3.8.16.3 Example
The following example shows all listeners that are currently defined.
wls:/mydomain/serverConfig> showListeners()

3.8.17 startRecording
Command Category: Information Commands
Use with WLST: Online or Offline

3.8.17.1 Description
Records all user interactions with WLST. This command is useful for capturing commands for replay.
In the event of an error, the command returns a WLSTException.
This command cannot be used when you are importing WLST as a Jython module, as described in "Importing WLST as a Jython Module" in Oracle WebLogic Scripting Tool.
**3.8.17.2 Syntax**

```
startRecording(recordFile, [recordAll])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>recordFile</td>
<td>Name of the file to which you want to record the WLST commands. The filename can be absolute or relative to the directory from which you invoked WLST.</td>
</tr>
<tr>
<td>recordAll</td>
<td>Optional. Boolean value specifying whether to capture all user interactions in the file. This argument defaults to false, indicating that only WLST commands are captured, and not WLST command output.</td>
</tr>
</tbody>
</table>

**3.8.17.3 Example**

The following example begins recording WLST commands in the `record.py` file:

```
startRecording('c:/myScripts/record.py')
```

```
Starting recording to c:/myScripts/record.py
```

**3.8.18 state**

Command Category: Information Commands

Use with WLST: Online

**3.8.18.1 Description**

Using Node Manager, returns a map of servers or clusters and their state. Node Manager must be running.

For more information about server states, see "Understanding Server Life Cycle" in *Managing Server Startup and Shutdown for Oracle WebLogic Server*.

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

**3.8.18.2 Syntax**

```
state(name, [type])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the server or cluster for which you want to retrieve the current state.</td>
</tr>
<tr>
<td>type</td>
<td>Optional. Type, Server or Cluster. This argument defaults to Server. When returning the state of a cluster, you must set this argument explicitly to Cluster, or the command will fail.</td>
</tr>
</tbody>
</table>

**3.8.18.3 Example**

The following example returns the state of the Managed Server, `managed1`.

```
state('managed1', 'Server')
```

```
Current state of 'managed1': SUSPENDED
```

The following example returns the state of the cluster, `mycluster`.

```
state('mycluster', 'Cluster')
```

```
There are 3 server(s) in cluster: mycluster
```

States of the servers are
3.8.19 stopRecording

Command Category: Information Commands
Use with WLST: Online or Offline

3.8.19.1 Description
Stops recording WLST commands. For information about starting a recording, see Section 3.8.17, "startRecording".
In the event of an error, the command returns a WLSTException.

3.8.19.2 Syntax
stopRecording()

3.8.19.3 Example
The following example stops recording WLST commands.

wls:/mydomain/serverConfig> stopRecording()
Stopping recording to c:\myScripts\record.py
wls:/mydomain/serverConfig>

3.8.20 stopRedirect

Command Category: Information Commands
Use with WLST: Online or Offline

3.8.20.1 Description
Stops the redirection of WLST output to a file, if redirection is in progress.
In the event of an error, the command returns a WLSTException.

3.8.20.2 Syntax
stopRedirect()

3.8.20.3 Example
The following example stops the redirection of WLST output to a file:

wls:/mydomain/serverConfig> stopRedirect()
WLST output will not be redirected to myfile.txt any more

3.8.21 storeUserConfig

Command Category: Information Commands
Use with WLST: Online
### 3.8.21.1 Description
Creates a user configuration file and an associated key file. The user configuration file contains an encrypted username and password. The key file contains a secret key that is used to encrypt and decrypt the username and password.

Only the key file that originally encrypted the username and password can be used to decrypt the values. If you lose the key file, you must create a new user configuration and key file pair.

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

### 3.8.21.2 Syntax
storeUserConfig([userConfigFile], [userKeyFile], [nm])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>userConfigFile</td>
<td>Optional. Name of the file to store the user configuration. The pathname can be absolute or relative to the file-system directory from which you started WLST. If you do not specify this argument, the command stores the file in your home directory as determined by your JVM. The location of the home directory depends on the SDK and type of operating system on which WLST is running. The default filename is based on the following pattern: <code>username-WebLogicConfig.properties</code> where <code>username</code> is the user name that you used to log in to the operating system. The command also prints to standard out the location in which it created the file.</td>
</tr>
<tr>
<td>userKeyFile</td>
<td>Optional. Name of the file to store the key information that is associated with the user configuration file that you specify. The pathname can be absolute or relative to the file-system directory from which you started WLST. If you do not specify this argument, the command stores the file in your home directory as determined by your JVM. The location of the home directory depends on the SDK and type of operating system on which WLST is running. The default filename is based on the following pattern: <code>username-WebLogicKey.properties</code> where <code>username</code> is the user name that you used to log in to the operating system. The command also prints to standard out the location in which it created the file.</td>
</tr>
<tr>
<td>nm</td>
<td>Optional. Boolean value specifying whether to store the username and password for Node Manager or WebLogic Server. If set to true, the Node Manager username and password is stored. This argument default to false.</td>
</tr>
</tbody>
</table>

### 3.8.21.3 Example
The following example creates and stores a user configuration file and key file in the default location.

```
$ wls:/mydomain/serverConfig> storeUserConfig()
Creating the key file can reduce the security of your system if it is not kept in a secured location after it is created. Do you want to create the key file? y or n  y
The username and password that were used for this current WLS connection are stored in C:\Documents and Settings\pat\pat-WebLogicConfig.properties
```
The following example creates and stores a user configuration file and key file in the specified locations.

```
wlst:/mydomain/serverConfig> storeUserConfig('c:/myFiles/myuserconfigfile.secure', 'c:/myFiles/myuserkeyfile.secure')
```

Creating the key file can reduce the security of your system if it is not kept in a secured location after it is created. Do you want to create the key file? y or n

```
y
```

The username and password that were used for this current WLS connection are stored in c:/myFiles/mysuserconfigfile.secure and c:/myFiles/myuserkeyfile.secure

### 3.8.22 threadDump

**Command Category:** Information Commands

**Use with WLST:** Online or Offline

#### 3.8.22.1 Description

Displays a thread dump for the specified server. In the event of an error, the command returns a WLSTException.

#### 3.8.22.2 Syntax

```
threadDump([writeToFile], [fileName], [serverName])
```

#### Argument Definition

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>writeToFile</td>
<td>Optional. Boolean value specifying whether to save the output to a file. This argument defaults to true, indicating that output is saved to a file.</td>
</tr>
<tr>
<td>fileName</td>
<td>Optional. Name of the file to which the output is written. The filename can be absolute or relative to the directory where WLST is running. This argument defaults to Thread_Dump_serverName file, where serverName indicates the name of the server. This argument is valid only if writeToFile is set to true.</td>
</tr>
<tr>
<td>serverName</td>
<td>Optional. Server name for which the thread dump is requested. This argument defaults to the server to which WLST is connected. If you are connected to an Administration Server, you can display a thread dump for the Administration Server and any Managed Server that is running in the WebLogic domain. If you are connected to a Managed Server, you can only display a thread dump for that Managed Server.</td>
</tr>
</tbody>
</table>

#### 3.8.22.3 Example

The following example displays the thread dump for the current server and saves the output to the Thread_Dump_serverName file.

```
wls:/mydomain/serverConfig> threadDump()
```

The following example displays the thread dump for the server managedServer. The information is not saved to a file.

```
wls:/mydomain/serverConfig> threadDump(writeToFile='false', serverName='managedServer')
```
3.8.23 viewMBean

Command Category: Information Commands

Use with WLST: Online

3.8.23.1 Description
Displays information about an MBean, such as the attribute names and values, and operations. In the event of an error, the command returns a WLSTException.

3.8.23.2 Syntax
viewMBean(mbean)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>mbean</td>
<td>MBean for which you want to display information.</td>
</tr>
</tbody>
</table>

3.8.23.3 Example
The following example displays information about the current MBean, cmo.

```
wlst:/mydomain/serverConfig> cmo.getType()
'Domain'
wlst:/mydomain/serverConfig> viewMBean(cmo)
Attribute Names and Values
--------------------------
XMLEntityCaches  null
Targets  javax.management.ObjectName[com.bea:
:Name=MedRecJMServer,Type=JMServer,
 :Name=WSSstoreForwardInternalJMServerMedRecServer,Type=JMServer,
 :Name=MedRecWseeJMServer,Type=JMServer,
 :Name=PhysWSEEJMServer,Type=JMServer,
 :Name=MedRecSAFAgent,Type=SAFAgent,
 :Name=AdminServer,Type=Server]
RootDirectory                                .
EmbeddedLDAP                          com.bea:Name=OOTB_medrec,Type=EmbeddedLDAP
RemoteSAFContexts  null
Libraries  javax.management.ObjectName[com.bea:
...  ...
wlst:/mydomain/serverConfig>
```

3.8.24 writeIniFile

Command Category: Information Commands

Use with WLST: Online

3.8.24.1 Description
Converts WLST definitions and method declarations to a Python (.py) file to enable advanced users to import them as a Jython module. After importing, the definitions and method declarations are available to other Jython modules and can be accessed directly using Jython syntax. For more information, see "Importing WLST as a Jython Module" in Oracle WebLogic Scripting Tool.

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

3.8.24.2 Syntax
writeIniFile(filePath)
3.8.24.3 Example

The following example converts WLST to a Python file named \texttt{wl.py}.

\texttt{wls:/offline> writeIniFile("wl.py")}

The Ini file is successfully written to \texttt{wl.py}
\texttt{wls:/offline>}

### 3.9 Life Cycle Commands

Use the WLST life cycle commands, listed in Table 3–10, to manage the life cycle of a server instance.

For more information about the life cycle of a server instance, see "Understanding Server Life Cycle" in \textit{Managing Server Startup and Shutdown for Oracle WebLogic Server}.

#### Table 3–10 Life Cycle Commands for WLST Configuration

<table>
<thead>
<tr>
<th>This command...</th>
<th>Enables you to...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>migrating</td>
<td>Migrate services to a target server within a cluster.</td>
<td>Online</td>
</tr>
<tr>
<td>resume</td>
<td>Resume a server instance that is suspended or in \texttt{ADMIN} state.</td>
<td>Online</td>
</tr>
<tr>
<td>shutdown</td>
<td>Gracefully shut down a running server instance or cluster.</td>
<td>Online</td>
</tr>
<tr>
<td>start</td>
<td>Start a Managed Server instance or a cluster using Node Manager.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td>startServer</td>
<td>Start the Administration Server.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td>suspend</td>
<td>Suspend a running server.</td>
<td>Online</td>
</tr>
</tbody>
</table>

#### 3.9.1 migrate

Command Category: Life Cycle Commands

Use with WLST: Online

#### 3.9.1.1 Description

Migrates the specified services (JTA, JMS, or Server) to a targeted server within a cluster. In the event of an error, the command returns a \texttt{WLSTException}.

For information about migrating services, see "Service Migration" in \textit{Using Clusters for Oracle WebLogic Server}.

#### 3.9.1.2 Syntax

\texttt{migrate\{sname, destinationName, [sourceDown], [destinationDown], [migrationType]\}}

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>sname</td>
<td>Name of the server from which the services should be migrated.</td>
</tr>
</tbody>
</table>
3.9.1.3 Example

The following example migrates all JMS and JTA services on server1 to the server server2. The boolean arguments specify that the source server is down and the destination server is running.

```
wls:/mydomain/edit !> migrate('server1','server2', 'true', 'false', 'all')
Migrating all JMS and JTA services from 'server1' to destination 'server2' ...
```

The following example migrates all Server services on server1 to the server server2. The boolean arguments specify that the source server is down and the destination server is running.

```
wls:/mydomain/edit !> migrate('server1','server2', 'true', 'false', 'Server')
Migrating singleton server services from 'server1' to machine 'server2'...
```

### 3.9.2 resume

#### 3.9.2.1 Description

Resumes a server instance that is suspended or in ADMIN state. This command moves a server to the RUNNING state. For more information about server states, see
"Understanding Server Life Cycle" in Managing Server Startup and Shutdown for Oracle WebLogic Server.

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

3.9.2.2 Syntax
resume([sname], [block])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>sname</td>
<td>Name of the server to resume. This argument defaults to the server to which WLST is currently connected.</td>
</tr>
<tr>
<td>block</td>
<td>Optional. Boolean value specifying whether WLST should block user interaction until the server is resumed. This argument defaults to false, indicating that user interaction is not blocked. In this case, WLST returns control to the user after issuing the command and assigns the task MBean associated with the current task to a variable that you can use to check its status. If you are importing WLST as a Jython module, as described in &quot;Importing WLST as a Jython Module&quot; in Oracle WebLogic Scripting Tool, block is always set to true.</td>
</tr>
</tbody>
</table>

3.9.3 Example
The following example resumes a Managed Server instance.

wls:/mydomain/serverConfig> resume('managed1', block='true')
Server 'managed1' resumed successfully.
wls:/mydomain/serverConfig>

3.9.3 shutdown

Command Category: Life Cycle Commands
Use with WLST: Online

3.9.3.1 Description
Gracefully shuts down a running server instance or a cluster. The shutdown command waits for all the in-process work to be completed before shutting down the server or cluster.

You shut down a server to which WLST is connected by entering the shutdown command without any arguments.

When connected to a Managed Server instance, you only use the shutdown command to shut down the Managed Server instance to which WLST is connected; you cannot shut down another server while connected to a Managed Server instance.

WLST uses Node Manager to shut down a Managed Server. When shutting down a Managed Server, Node Manager must be running.

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

3.9.3.2 Syntax
shutdown([name], [entityType], [ignoreSessions], [timeOut], [force], [block])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Optional. Name of the server or cluster to shutdown. This argument defaults to the server to which WLST is currently connected.</td>
</tr>
</tbody>
</table>
3.9.3.3 Example

The following example instructs WLST to shutdown the server to which you are connected:

```bash
wls:/mydomain/serverConfig> shutdown()
Shutting down the admin server that you are currently connected to .......
Disconnected from weblogic server: AdminServer
```

The following example instructs WLST to wait 1000 seconds for HTTP sessions to complete or timeout (at 1000 seconds) before shutting down myserver:

```bash
wls:/mydomain/serverConfig> shutdown('myserver','Server','false',1000,
block='false')
```

The following example instructs WLST to drop all HTTP sessions immediately while connected to a Managed Server instance:

```bash
wls:/mydomain/serverConfig> shutdown('MServer1','Server','true',1200)
Shutting down a managed server that you are connected to ...
Disconnected from weblogic server: MServer1
```

The following example instructs WLST to shutdown the cluster mycluster:

```bash
wls:/mydomain/serverConfig> shutdown('mycluster','Cluster')
Shutting down the cluster with name mycluster
Shutdown of cluster mycluster has been issued, please refer to the logs to check if the cluster shutdown is successful.
Use the state(<server-name>) or state(<cluster-name>,"Cluster") to check the status of the server or cluster
wls:/mydomain/serverConfig> state('mycluster','Cluster')
There are 3 server(s) in cluster: mycluster
```

States of the servers are
3.9.4 start

Command Category: Life Cycle Commands

Use with WLST: Online

3.9.4.1 Description

Starts a Managed Server instance or a cluster using Node Manager. WLST must be connected to the Administration Server and Node Manager must be running.

For more information about WLST commands used to connect to and use Node Manager, see Section 3.10, "Node Manager Commands".

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

3.9.4.2 Syntax

start(name, [type], [url], [block])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the Managed Server or cluster to start.</td>
</tr>
<tr>
<td>type</td>
<td>Optional. Type, Server or Cluster. This argument defaults to Server. When starting a cluster, you must set this argument explicitly to Cluster, or the command will fail.</td>
</tr>
<tr>
<td>url</td>
<td>Optional. Listen address and listen port of the server instance, specified using the following format: [protocol://]listen-address:listen-port. If not specified, this argument defaults to t3://localhost:7001.</td>
</tr>
<tr>
<td>block</td>
<td>Optional. Boolean value specifying whether WLST should block user interaction until the server or cluster is started. This argument defaults to false, indicating that user interaction is not blocked. In this case, WLST returns control to the user after issuing the command and assigns the task MBean associated with the current task to a variable that you can use to check its status. If you are importing WLST as a Jython module, as described &quot;Importing WLST as a Jython Module&quot; in Oracle WebLogic Scripting Tool, block is always set to true.</td>
</tr>
</tbody>
</table>

3.9.4.3 Example

The following example instructs Node Manager to start a Managed Server instance; the listen address is localhost and listen port is 8801. WLST returns control to the user after issuing this command, as block is set to false.

```
wlst:/mydomain/serverConfig> start('myserver', 'Server', block='false')
Starting server myserver ...  
Server with name myserver started successfully.
wlst:/mydomain/serverConfig>
```

The following example instructs Node Manager to start a cluster. WLST block user interaction until the cluster is started, as block defaults to true.

```
wlst:/mydomain/serverConfig> start('mycluster', 'Cluster')  
Starting the following servers in Cluster, mycluster: MS1, MS2, MS3...
```

MServer1---SHUTDOWN
MServer2---SHUTDOWN
MServer3---SHUTDOWN
All servers in the cluster mycluster are started successfully.

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

### 3.9.5 startServer

**Command Category:** Life Cycle Commands  

**Use with WLST:** Online or Offline  

#### 3.9.5.1 Description

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

---

**Note:** You can use `startServer` only to start a WebLogic Administration Server, by running WLST from the WL_HOME/common/bin directory. You cannot use `startServer` to start an integrated WebLogic Administration Server (that is, an Administration Server for a Fusion Middleware Suite product installed in an ORACLE_HOME directory).

To start the Administration server for a Fusion Middleware Suite product other than WebLogic Server, use either of the following methods:

- Execute the server startup script for the associated WebLogic domain.
- Start the server using Node Manager. If you use this method, make sure that the `startScriptEnabled` property is set to true in Node Manager.

---

#### 3.9.5.2 Syntax

```java
startServer([adminServerName], [domainName], [url], [username], [password], [domainDir], [block], [timeout], [serverLog], [systemProperties], [jvmArgs], [spaceAsJvmArgsDelimiter])
```

**Argument Definition**

- `adminServerName`  
  Optional. Name of the Administration Server to start. This argument defaults to `myserver`.

- `domainName`  
  Optional. Name of the WebLogic domain to which the Administration Server belongs. This argument defaults to `mydomain`.

- `url`  
  Optional. URL of the Administration Server. The URL supplied with the `startServer` command will override the listen address and port specified in the config.xml file. If not specified on the command line or in the `config.xml` file, this argument defaults to `t3://localhost:7001`.

- `username`  
  Optional. Username use to connect WLST to the server. This argument defaults to `weblogic`.

- `password`  
  Optional. Password used to connect WLST to the server. This argument defaults to `welcome1`.

- `domainDir`  
  Optional. Domain directory in which the Administration Server is being started. This argument defaults to the directory from which you started WLST.
3.9.5.3 Example
The following example starts the Administration Server named `demoServer` in the `demoDomain`.

```
wlst:/offline> startServer('demoServer','demoDomain','t3://localhost:8001',
'myweblogic','wlstdomain','c:/mydomains/wlst','false',
60000,
jvmArgs='-XX:MaxPermSize=75m, -Xmx512m, -XX:+UseParallelGC')
wlst:/offline>
```

3.9.6 suspend
Command Category: Life Cycle Commands
Use with WLST: Online

3.9.6.1 Description
Suspends a running server. This command moves a server from the RUNNING state to the ADMIN state. For more information about server states, see "Understanding Server Life Cycle" in Managing Server Startup and Shutdown for Oracle WebLogic Server.

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

3.9.6.2 Syntax
```
suspend([sname], [ignoreSessions], [timeOut], [force], [block])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>sname</td>
<td>Optional. Name of the server to suspend. The argument defaults to the server to which WLST is currently connected.</td>
</tr>
<tr>
<td>ignoreSessions</td>
<td>Optional. Boolean value specifying whether WLST should drop all HTTP sessions immediately or wait for HTTP sessions to complete or time out while suspending. This argument defaults to false, indicating that HTTP sessions must complete or time out.</td>
</tr>
</tbody>
</table>
3.9.6.3 Example
The following example suspends a Managed Server instance:

```
wls:/mydomain/serverConfig> suspend('managed1')
Server 'managed1' suspended successfully.
wls:/mydomain/serverConfig>
```

3.10 Node Manager Commands
Use the WLST Node Managers commands, listed in Table 3–11, to start, shut down, restart, and monitor WebLogic Server instances.

**Node Manager must be running before you can execute the commands within this category.**

For more information about Node Manager, see "Using Node Manager" in the Node Manager Administrator’s Guide for Oracle WebLogic Server.

Table 3–11  Node Manager Commands for WLST Configuration

<table>
<thead>
<tr>
<th>This command...</th>
<th>Enables you to...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>nm</code></td>
<td>Determine whether WLST is connected to Node Manager.</td>
<td>Online</td>
</tr>
<tr>
<td><code>nmConnect</code></td>
<td>Connect WLST to Node Manager to establish a session.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td><code>nmDisconnect</code></td>
<td>Disconnect WLST from a Node Manager session.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td><code>nmEnroll</code></td>
<td>Enables the Node Manager on the current computer to manage servers in a specified WebLogic domain.</td>
<td>Online</td>
</tr>
<tr>
<td><code>nmGenBootStartupProps</code></td>
<td>Generates the Node Manager property files, <code>boot.properties</code> and <code>startup.properties</code>, for the specified server.</td>
<td>Online</td>
</tr>
<tr>
<td><code>nmKill</code></td>
<td>Kill the specified server instance that was started with Node Manager.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td><code>nmLog</code></td>
<td>Return the Node Manager log.</td>
<td>Online or Offline</td>
</tr>
</tbody>
</table>
### 3.10.1 nm

**Command Category:** Node Manager Commands  
**Use with WLST:** Online or Offline

**3.10.1.1 Description**

Determines whether WLST is connected to Node Manager. Returns `true` or `false` and prints a descriptive message. Node Manager must be running before you can execute this command.

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

**3.10.1.2 Syntax**

```
nm()
```

**3.10.1.3 Example**

The following example indicates that WLST is currently connected to Node Manager that is monitoring `mydomain`.

```
wls:/mydomain/serverConfig> nm()
Currently connected to Node Manager that is monitoring the domain "mydomain"
wls:/mydomain/serverConfig>
```

The following example indicates that WLST is not currently connected to Node Manager.

```
wls:/mydomain/serverConfig> nm()
Not connected to any Node Manager
wls:/mydomain/serverConfig>
```

### 3.10.2 nmConnect

**Command Category:** Node Manager Commands  
**Use with WLST:** Online or Offline

**3.10.2.1 Description**

Connects WLST to Node Manager to establish a session. After connecting to Node Manager, you can invoke any Node Manager commands via WLST. Node Manager must be running before you can execute this command.
Once connected, the WLST prompt displays as follows, where `domainName` indicates the name of the WebLogic domain that is being managed: `wls:/nm/domainName>`. If you then connect WLST to a WebLogic Server instance, the prompt is changed to reflect the WebLogic Server instance. You can use the `nm` command to determine whether WLST is connected to Node Manager, as described in Section 3.10.1, "nm".

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

### 3.10.2.2 Syntax

```
nmConnect([username, password], [host], [port], [domainName], [domainDir] [nmType], [verbose])
nmConnect([userConfigFile, userKeyFile], [host], [port], [domainName], [domainDir] [nmType], [verbose])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>username</code></td>
<td>Username of the operator who is connecting WLST to Node Manager. The username defaults to <code>weblogic</code>.</td>
</tr>
<tr>
<td><strong>Note:</strong> When running a server in production mode, you must specify the username and password explicitly on the command line to ensure that the appropriate username and password are used when connecting to Node Manager.</td>
<td></td>
</tr>
<tr>
<td><code>password</code></td>
<td>Password of the operator who is connecting WLST to Node Manager. The password defaults to <code>welcome1</code>.</td>
</tr>
<tr>
<td><strong>Note:</strong> When running a server in production mode, you must specify the username and password explicitly on the command line to ensure that the appropriate username and password are used when connecting to Node Manager.</td>
<td></td>
</tr>
<tr>
<td><code>host</code></td>
<td>Optional. Host name of Node Manager. This argument defaults to localhost.</td>
</tr>
<tr>
<td><code>port</code></td>
<td>Optional. Port number of Node Manager. This argument defaults to a value that is based on the Node Manager server type, as follows:</td>
</tr>
<tr>
<td>■ For plain type, defaults to 5556</td>
<td></td>
</tr>
<tr>
<td>■ For rsh type, defaults to 514</td>
<td></td>
</tr>
<tr>
<td>■ For ssh type, defaults to 22</td>
<td></td>
</tr>
<tr>
<td>■ For ssl type, defaults to 5556</td>
<td></td>
</tr>
<tr>
<td><code>domainName</code></td>
<td>Optional. Name of the WebLogic domain that you want to manage. This argument defaults to mydomain.</td>
</tr>
<tr>
<td><code>domainDir</code></td>
<td>Optional. Path of the domain directory to which you want to save the Node Manager secret file (<code>nm_password.properties</code>) and <code>SerializedSystemIni.dat</code> file. This argument defaults to the directory in which WLST was started.</td>
</tr>
<tr>
<td><code>nmType</code></td>
<td>Type of the Node Manager server. Valid values include:</td>
</tr>
<tr>
<td>■ plain for plain socket Java-based implementation</td>
<td></td>
</tr>
<tr>
<td>■ rsh for RSH implementation</td>
<td></td>
</tr>
<tr>
<td>■ ssh for script-based SSH implementation</td>
<td></td>
</tr>
<tr>
<td>■ ssl for Java-based SSL implementation</td>
<td></td>
</tr>
<tr>
<td>This argument defaults to <code>ssl</code>.</td>
<td></td>
</tr>
<tr>
<td><code>verbose</code></td>
<td>Optional. Boolean value specifying whether WLST connects to Node Manager in verbose mode. This argument defaults to <code>false</code>, disabling verbose mode.</td>
</tr>
</tbody>
</table>
### 3.10.2.3 Example

The following example connects WLST to Node Manager to monitor the oamdomain domain using the default host and port numbers and plain Node Manager type.

```bash
wls:/myserver/serverConfig> nmConnect('weblogic', 'welcome1', 'localhost', '5555', 'oamdomain', 'c:/Oracle/Middleware/user_projects/domains/oamdomain','plain')
Connecting to Node Manager Server ...
Successfully connected to Node Manager.
```

The following example connects WLST to a Node Manager Server instance using a user configuration and key file to provide user credentials.

```bash
wls:/myserver/serverConfig> nmConnect(userConfigFile='c:/myfiles/myuserconfigfile.secure', userKeyFile='c:/myfiles/myuserkeyfile.secure', host='172.18.137.82', port=26106, domainName='mydomain', domainDir='c:/myfiles/mydomain', mType='plain')
Connecting to Node Manager Server ...
Successfully connected to Node Manager.
```

### 3.10.3 nmDisconnect

**Command Category:** Node Manager Commands

**Use with WLST:** Online or Offline

WLST must be connected to Node Manager to run this command.

#### 3.10.3.1 Description

Disconnects WLST from a Node Manager session.

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

#### 3.10.3.2 Syntax

`nmDisconnect()`

#### 3.10.3.3 Example

The following example disconnects WLST from a Node Manager session.

```bash
wls:/nm/oamdomain> nmDisconnect()
Successfully disconnected from Node Manager
wls:/myserver/serverConfig>
```

---

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>userConfigFile</td>
<td>Optional. Name and location of a user configuration file which contains an encrypted username and password. When you create a user configuration file, the <code>storeUserConfig</code> command uses a key file to encrypt the username and password. Only the key file that encrypts a user configuration file can decrypt the username and password. (See Section 3.8.21, &quot;storeUserConfig&quot;).</td>
</tr>
<tr>
<td>userKeyFile</td>
<td>Optional. Name and location of the key file that is associated with the specified user configuration file and is used to decrypt it. (See Section 3.8.21, &quot;storeUserConfig&quot;).</td>
</tr>
</tbody>
</table>
3.10.4 nmEnroll

Command Category: Node Manager Commands

Use with WLST: Online

3.10.4.1 Description

Enrolls the machine on which WLST is currently running. WLST must be connected to an Administration Server to run this command; WLST does not need to be connected to Node Manager.

This command downloads the following files from the Administration Server:

- Node Manager secret file (nm_password.properties), which contains the encrypted username and password that is used for server authentication
- SerializedSystemIni.dat file

This command also updates the nodemanager.domains file under the WL_HOME/common/nodemanager directory with the domain information, where WL_HOME refers to the top-level installation directory for WebLogic Server.

You must run this command once per WebLogic domain per machine unless that domain shares the root directory of the Administration Server.

If the machine is already enrolled when you run this command, the Node Manager secret file (nm_password.properties) is refreshed with the latest information from the Administration Server.

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

3.10.4.2 Syntax

nmEnroll([domainDir], [nmHome])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>domainDir</td>
<td>Optional. Path of the domain directory to which you want to save the Node Manager secret file (nm_password.properties) and SerializedSystemIni.dat file. This argument defaults to the directory in which WLST was started.</td>
</tr>
<tr>
<td>nmHome</td>
<td>Optional. Path to the Node Manager home. The nodemanager.domains file, containing the domain information, is written to this directory. This argument defaults to WL_HOME/common/nodemanager, where WL_HOME refers to the top-level installation directory for WebLogic Server.</td>
</tr>
</tbody>
</table>

3.10.4.3 Example

The following example enrolls the current machine with Node Manager and saves the Node Manager secret file (nm_password.properties) and SerializedSystemIni.dat file to c:/Oracle/Middleware/mydomain/common/nodemanager/nm_password.properties. The nodemanager.domains file is written to WL_HOME/common/nodemanager by default.

wls:/mydomain/serverConfig>

nmEnroll('c:/Oracle/Middleware/mydomain/common/nodemanager')

Enrolling this machine with the domain directory at c:\Oracle\Middleware\mydomain\common\nodemanager....

Successfully enrolled this machine with the domain directory at C:\Oracle\Middleware\mydomain\common\nodemanager

wls:/mydomain/serverConfig>
3.10.5 nmGenBootStartupProps

Command Category: Node Manager Commands

Use with WLST: Online

3.10.5.1 Description
Generates the Node Manager property files, boot.properties and startup.properties, for the specified server. The Node Manager property files are stored relative to the root directory of the specified server. The target root directory must be on the same machine on which you are running the command.

You must specify the name of a server; otherwise, the command will fail.

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

3.10.5.2 Syntax

nmGenBootStartupProps(serverName)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>serverName</td>
<td>Name of the server for which Node Manager property files are generated.</td>
</tr>
</tbody>
</table>

3.10.5.3 Example

The following example generates boot.properties and startup.properties in the root directory of the specified server, ms1.

wls:/mydomain/serverConfig> nmGenBootStartupProps('ms1')
Successfully generated boot.properties at
C:\Oracle\Middleware\mydomain\servers\ms1\data\nodemanager\boot.properties
Successfully generated startup.properties at
C:\Oracle\Middleware\mydomain\servers\ms1\data\nodemanager\startup.properties
wls:/mydomain/serverConfig>

3.10.6 nmKill

Command Category: Node Manager Commands

Use with WLST: Online or Offline

WLST must be connected to Node Manager to run this command.

3.10.6.1 Description
Kills the specified server instance that was started with Node Manager.

If you do not specify a server name using the serverName argument, the argument defaults to myServer, which must match your server name or the command will fail.

If you attempt to kill a server instance that was not started using Node Manager, the command displays an error.

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

3.10.6.2 Syntax

nmKill([serverName])
3.10.6.3 Example
The following example kills the server named oamserver.

```bash
wls:/nm/oamdomain> nmKill('oamserver')
Killing server 'oamserver' ...
Server oamServer killed successfully.
wls:/nm/oamdomain>
```

3.10.7 nmLog
Command Category: Node Manager Commands
Use with WLST: Online or Offline
WLST must be connected to Node Manager to run this command.

3.10.7.1 Description
Returns the Node Manager log.
In the event of an error, the command returns a `WLSTException`.

3.10.7.2 Syntax
`nmLog([writer])`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>writer</td>
<td>Optional <code>java.io.Writer</code> object to which you want to stream the log output. This argument defaults to the WLST writer stream.</td>
</tr>
</tbody>
</table>

3.10.7.3 Example
The following example displays the Node Manager log.

```bash
wls:/nm/oamdomain> nmLog()
Successfully retrieved the Node Manager log and written.
wls:/nm/oamdomain>
```

3.10.8 nmServerLog
Command Category: Node Manager Commands
Use with WLST: Online or Offline
WLST must be connected to Node Manager to run this command.

3.10.8.1 Description
Returns the server output log of the server that was started with Node Manager.
In the event of an error, the command returns a `WLSTException`.

3.10.8.2 Syntax
`nmServerLog([serverName], [writer])`
3.10.8.3 Example

The following example displays the server output log for the `oamserver` server and writes the log output to `myWriter`.

```wls
wls:/nm/oamdomain> nmServerLog('oamserver',myWriter)
Successfully retrieved the server log and written.
wls:/nm/oamdomain>
```

3.10.9 `nmServerStatus`

**Command Category:** Node Manager Commands

**Use with WLST:** Online or Offline

WLST must be connected to Node Manager to run this command.

3.10.9.1 Description

Returns the status of the server that was started with Node Manager.

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

3.10.9.2 Syntax

```java
nmServerStatus([serverName])
```

3.10.9.3 Example

The following example displays the status of the server named `oamserver`, which was started with Node Manager.

```wls
wls:/nm/oamdomain> nmServerStatus('oamserver')
RUNNING
wls:/nm/oamdomain>
```

3.10.10 `nmStart`

**Command Category:** Node Manager Commands

**Use with WLST:** Online or Offline

WLST must be connected to Node Manager to run this command.

3.10.10.1 Description

Starts a server in the current WebLogic domain using Node Manager.

In the event of an error, the command returns a `WLSTException`. 
### Node Manager Commands

#### 3.10.10 nmStart

**Syntax**

nmStart([serverName], [domainDir], [props], [writer])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>serverName</td>
<td>Optional. Name of the server to be started.</td>
</tr>
<tr>
<td>domainDir</td>
<td>Optional. Domain directory of the server to be started. This argument</td>
</tr>
<tr>
<td></td>
<td>defaults to the directory from which you started WLST.</td>
</tr>
<tr>
<td>props</td>
<td>Optional. System properties to apply to the new server.</td>
</tr>
<tr>
<td>writer</td>
<td>Optional. java.io.Writer object to which the server output is written.</td>
</tr>
<tr>
<td></td>
<td>This argument defaults to the WLST writer.</td>
</tr>
</tbody>
</table>

**Example**

The following example starts the managed1 server in the current WebLogic domain using Node Manager.

```wls
wls:/nm/mydomain> nmStart("managed1")
Starting server managed1 ....
Server managed1 started successfully
wls:/nm/mydomain>
```

The following example starts the Administration Server in the specified WebLogic domain using Node Manager. In this example, the props variable stores the system property settings and is passed to the command using the props argument.

```wls
wls:/nm/mydomain> prps = makePropertiesObject("weblogic.ListenPort=8001")
wls:/nm/mydomain> nmStart("AdminServer", props=prps)
Starting server AdminServer ...
Server AdminServer started successfully
wls:/nm/mydomain>
```

#### 3.10.11 nmVersion

**Command Category:** Node Manager Commands

**Use with WLST:** Online or Offline

WLST must be connected to Node Manager to run this command.

**Description**

Returns the Node Manager version.

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

**Syntax**

nmVersion()

**Example**

The following example displays the Node Manager version.
3.10.12 startNodeManager

Command Category: Node Manager Commands
Use with WLST: Online or Offline

3.10.12.1 Description
Starts Node Manager on the same computer that is running WLST.

Note: The WebLogic Server custom installation process optionally installs and starts Node Manager as a Windows service on Windows systems. For more information, see "About Installing Node Manager as a Windows Service" in the Oracle WebLogic Server Installation Guide. In this case, you do not need to start the Node Manager manually.

If Node Manager is already running when you invoke the startNodeManager command, the following message is displayed:

A Node Manager has already been started.
Cannot start another Node Manager process via WLST

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

3.10.12.2 Syntax
startNodeManager([verbose], [nmProperties])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>verbose</td>
<td>Optional. Boolean value specifying whether WLST starts Node Manager in verbose mode. This argument defaults to false, disabling verbose mode.</td>
</tr>
<tr>
<td>nmProperties</td>
<td>Optional. Comma-separated list of Node Manager properties, specified as name-value pairs. Node Manager properties include, but are not limited to, the following: NodeManagerHome, ListenAddress, ListenPort, and PropertiesFile.</td>
</tr>
</tbody>
</table>

3.10.12.3 Example
The following example displays the Node Manager server version.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>verbose</td>
<td>Optional. Boolean value specifying whether WLST starts Node Manager in verbose mode. This argument defaults to false, disabling verbose mode.</td>
</tr>
<tr>
<td>nmProperties</td>
<td>Optional. Comma-separated list of Node Manager properties, specified as name-value pairs. Node Manager properties include, but are not limited to, the following: NodeManagerHome, ListenAddress, ListenPort, and PropertiesFile.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>verbose</td>
<td>Optional. Boolean value specifying whether WLST starts Node Manager in verbose mode. This argument defaults to false, disabling verbose mode.</td>
</tr>
<tr>
<td>nmProperties</td>
<td>Optional. Comma-separated list of Node Manager properties, specified as name-value pairs. Node Manager properties include, but are not limited to, the following: NodeManagerHome, ListenAddress, ListenPort, and PropertiesFile.</td>
</tr>
</tbody>
</table>

The Node Manager process is running independent of the WLST process Exiting WLST will not stop the Node Manager process. Please refer to the Node Manager logs for more information. The Node Manager logs will be under c:\Oracle\Middleware\wlserver_10.3\common\nodemanager.
3.11 Tree Commands

Use the WLST tree commands, listed in Table 3–12, to navigate among MBean hierarchies.

### Table 3–12 Tree Commands for WLST Configuration

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>custom</td>
<td>Navigate to the root of custom MBeans that are registered in the server.</td>
<td>Online</td>
</tr>
<tr>
<td>domainConfig</td>
<td>Navigate to the last MBean to which you navigated in the domain configuration hierarchy or to the root of the hierarchy, DomainMBean.</td>
<td>Online</td>
</tr>
<tr>
<td>domainCustom</td>
<td>Navigate to the root of custom MBeans that are registered in the Domain Runtime MBean Server</td>
<td>Online</td>
</tr>
<tr>
<td>domainRuntime</td>
<td>Navigate to the last MBean to which you navigated in the domain runtime hierarchy or to the root of the hierarchy, DomainRuntimeMBean.</td>
<td>Online</td>
</tr>
<tr>
<td>edit</td>
<td>Navigate to the last MBean to which you navigated in the edit configuration MBean hierarchy or to the root of the hierarchy, DomainMBean.</td>
<td>Online</td>
</tr>
<tr>
<td>jndi</td>
<td>Navigates to the JNDI tree for the server to which WLST is currently connected.</td>
<td>Online</td>
</tr>
<tr>
<td>serverConfig</td>
<td>Navigate to the last MBean to which you navigated in the configuration MBean hierarchy or to the root of the hierarchy, DomainMBean.</td>
<td>Online</td>
</tr>
<tr>
<td>serverRuntime</td>
<td>Navigate to the last MBean to which you navigated in the runtime MBean hierarchy or to the root of the hierarchy, ServerRuntimeMBean.</td>
<td>Online</td>
</tr>
</tbody>
</table>

3.11.1 custom

Command Category: Tree Commands

Use with WLST: Online

#### 3.11.1.1 Description

Navigates to the root of custom MBeans that are registered in the Runtime MBean Server. WLST navigates, interrogates, and edits custom MBeans as it does domain MBeans; however, custom MBeans cannot use the cmo variable because a stub is not available.

---

**Note:** When navigating to the custom tree, WLST queries all MBeans in the compatibility MBean server, the runtime MBean server, and potentially the JVM platform MBean server to locate the custom MBeans. Depending on the number of MBeans in the current WebLogic domain, this process may take a few minutes, and WLST may not return a prompt right away.

---

The custom command is available when WLST is connected to an Administration Server instance or a Managed Server instance. When connected to a WebLogic Integration or WebLogic Portal server, WLST can interact with all the WebLogic Integration or WebLogic Portal server MBeans.
For more information about custom MBeans, see *Developing Custom Management Utilities With JMX for Oracle WebLogic Server*.

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

---

**Note:** You can also navigate to custom MBeans on the Domain Runtime MBean Server using the `domainCustom()` command. See Section 3.11.3, "domainCustom," for more information.

---

### 3.11.2 domainConfig

**Command Category:** Tree Commands  
**Use with WLST:** Online

#### 3.11.2.1 Description

Navigates to the last MBean to which you navigated in the domain Configuration hierarchy or to the root of the hierarchy, `DomainMBean`. This read-only hierarchy stores the configuration MBeans that represent your current WebLogic domain.

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

#### 3.11.2.2 Syntax

`domainConfig()`

#### 3.11.2.3 Example

The following example navigates from the configuration MBean hierarchy to the WebLogic domain Configuration hierarchy on an Administration Server instance.

```text
wls:/mydomain/serverConfig> domainConfig()  
Location changed to domainConfig tree. This is a read-only tree with `DomainMBean` as the root.  
For more help, use help('domainConfig')

wls:/mydomain/domainConfig>
```

---

### 3.11.1 custom()

**Syntax**

`custom()`

**Example**

The following example navigates from the configuration MBean hierarchy to the custom MBean hierarchy on a Administration Server instance.

```text
wls:/mydomain/serverConfig> custom()  
Location changed to custom tree. This is a writeable tree with No root. For more help, use help('custom')

wls:/mydomain/custom>
```
3.11.3 domainCustom

Command Category: Tree Commands

Use with WLST: Online

3.11.3.1 Description

Navigates to the domain custom tree of custom MBeans that are registered in the Domain Runtime MBean Server. WLST navigates, interrogates, and edits domain custom MBeans as it does domain MBeans; however, domain custom MBeans cannot use the cmo variable because a stub is not available.

Note: When navigating to the domainCustom tree, WLST queries all MBeans in the Domain Runtime MBean Server, the Runtime MBean Servers on each server, and potentially the JVM platform MBean server to locate the custom MBeans. Depending on the number of MBeans in the current WebLogic domain, this process make take a few minutes, and WLST may not return a prompt right away. It is recommended that a JMX query Object Name Pattern be specified to limit the amount of searching performed.

The domainCustom command is available only when WLST is connected to an Administration Server instance.

For more information about the Domain Runtime MBean Server, see "Understanding WebLogic Server MBeans" in Developing Custom Management Utilities With JMX for Oracle WebLogic Server.

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

3.11.3.2 Syntax

domainCustom(ObjectNamePattern)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ObjectNamePattern</td>
<td>A JMX query pattern, such as sip:<em>.</em>. The default value is null or <em>:</em>.*.</td>
</tr>
</tbody>
</table>

3.11.3.3 Example

The following example navigates from the configuration MBean hierarchy to the domain custom MBean hierarchy on an Administration Server instance:

```
wlst:/mydomain/serverConfig> domainCustom()
Location changed to domain custom tree. This is a writeable tree with No root. For more help, use help('domainCustom').
```
3.11.4 domainRuntime

Command Category: Tree Commands
Use with WLST: Online

3.11.4.1 Description
Navigates to the last MBean to which you navigated in the domain Runtime hierarchy or to the root of the hierarchy, DomainRuntimeMBean. This read-only hierarchy stores the runtime MBeans that represent your current WebLogic domain.

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

3.11.4.2 Syntax
domainRuntime()

3.11.4.3 Example
The following example navigates from the configuration MBean hierarchy to the domain Runtime hierarchy on an Administration Server instance.

```
wls:/mydomain/serverConfig> domainRuntime()
wls:/mydomain/domainRuntime>
ls()
dr-- AppRuntimeStateRuntime
dr-- DeployerRuntime
dr-- DomainServices
dr-- LogRuntime
dr-- MessageDrivenControlEJBRuntime
dr-- MigratableServiceCoordinatorRuntime
dr-- MigrationDataRuntimes
dr-- SNMPAgentRuntime
dr-- ServerLifeCycleRuntimes
dr-- ServerRuntimes
dr-- ServerServices
```

```
-r-- ActivationTime Mon Aug 01 11:41:25 EDT 2005
-r-- Clusters null
-r-- MigrationDataRuntimes null
-r-- Name sampleMedRecDomain
-rw- Parent null
-r-- SNMPAgentRuntime null
-r-- Type DomainRuntime
-r-x restartSystemResource
    WebLogicMBean(weblogic.management.configuration.SystemResourceMBean)
```

```
wls:/mydomain/domainRuntime>
```

3.11.5 edit

Command Category: Tree Commands
Use with WLST: Online

3.11.5.1 Description
Navigates to the last MBean to which you navigated in the edit configuration MBean hierarchy or to the root of the hierarchy, DomainMBean. This writable hierarchy stores all of the configuration MBeans that represent your current WebLogic domain.
In the event of an error, the command returns a WLSTException.

3.11.5.2 Syntax

edit()

3.11.5.3 Example

The following example illustrates how to navigate from the server configuration MBean hierarchy to the editable copy of the domain configuration MBean hierarchy, in an Administration Server instance.

wls:/myserver/serverConfig> edit()
Location changed to edit tree. This is a writeable tree with DomainMBean as the root.
For more help, use help('edit')
wls:/myserver/edit !>
ls()
dr-- AppDeployments
dr-- BridgeDestinations
dr-- Clusters
dr-- DeploymentConfiguration
dr-- Deployments
dr-- EmbeddedLDAP
...
wls:/myserver/edit !>

3.11.6 jndi

Command Category: Tree Commands
Use with WLST: Online

3.11.6.1 Description

Navigates to the JNDI tree for the server to which WLST is currently connected. This read-only tree holds all the elements that are currently bound in JNDI.

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

3.11.6.2 Syntax

jndi()
3.11.6.3 Example
The following example navigates from the runtime MBean hierarchy to the Domain JNDI tree on an Administration Server instance.

```
wlst:/myserver/runtime> jndi()
Location changed to jndi tree. This is a read-only tree with No root. For more help, use help('jndi')
wlst:/myserver/jndi> ls()
dr-- ejb
dr-- javax
dr-- jms
dr-- weblogic
...
```

3.11.7 serverConfig
Command Category: Tree Commands
Use with WLST: Online

3.11.7.1 Description
Navigates to the last MBean to which you navigated in the configuration MBean hierarchy or to the root of the hierarchy, DomainMBean.

This read-only hierarchy stores the configuration MBeans that represent the server to which WLST is currently connected. The MBean attribute values include any command-line overrides that a user specified while starting the server.

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

For more information, see "Navigating Among MBean Hierarchies" in Oracle WebLogic Scripting Tool.

3.11.7.2 Syntax
```
serverConfig()
```

3.11.7.3 Example
The following example navigates from the domain runtime MBean hierarchy to the configuration MBean hierarchy on an Administration Server instance.

```
wls:/mydomain/domainRuntime> serverConfig()
wls:/mydomain/serverConfig>
```

3.11.8 serverRuntime
Command Category: Tree Commands
Use with WLST: Online

3.11.8.1 Description
Navigates to the last MBean to which you navigated in the runtime MBean hierarchy or to the root of the hierarchy, ServerRuntimeMBean. This read-only hierarchy stores the runtime MBeans that represent the server to which WLST is currently connected.

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

serverRuntime()

3.11.8.3 Example

The following example navigates from the configuration MBean hierarchy to the
runtime MBean hierarchy on an Administration Server instance.

wls:/mydomain/serverConfig> serverRuntime()
Location changed to serverRuntime tree. This is a read-only tree with
ServerRuntimeMBean as the root.
For more help, use help('serverRuntime')
wls:/mydomain/serverRuntime>

3.12 WLST Variable Reference

Table 3-13 describes WLST variables and their common usage. All variables are
initialized to default values at the start of a user session and are changed according to
the user interaction with WLST.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
</table>
| cmo            | Current Management Object. The cmo variable is set to the bean instance to which you navigate using WLST. You can use this variable to perform any get, set, or invoke method on the current bean instance. WLST sets the variable to the current WLST path. For example, when you change to the serverConfig hierarchy, `cmo` is set to DomainMBean. When you change to the serverRuntime hierarchy, `cmo` is set to ServerRuntimeMBean. The variable is available in all WLST hierarchies except custom and jndi. | wls:/mydomain/edit>
cmo.setAdministrationPort(9092) |
| connected      | Boolean value specifying whether WLST is connected to a running server. WLST sets this variable to true when connected to a running server; otherwise, WLST sets it to false. | wls:/mydomain/serverConfig> print connected
false                      |
| domainName     | Name of the WebLogic domain to which WLST is connected.                      | wls:/mydomain/serverConfig> print domainName
mydomain               |
| domainRuntimeService | DomainRuntimeServiceMBean MBean. This variable is available only when WLST is connected to the Administration Server. | wls:/mydomain/serverConfig> domainService.getServerName()
'myserver'             |
| editService    | EditServiceMBean MBean. This variable is available only when WLST is connected to the Administration Server. | wls:/mydomain/edit> dc =
editservice.getDomainConfiguration() |
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>exitonerror</td>
<td>Boolean value specifying whether WLST terminates script execution when it</td>
<td>wls:/mydomain/serverConfig&gt; print exitonerror true</td>
</tr>
<tr>
<td></td>
<td>encounters an exception. This variable defaults to true, indicating that</td>
<td></td>
</tr>
<tr>
<td></td>
<td>script execution is terminated when WLST encounters an error. This variable</td>
<td></td>
</tr>
<tr>
<td></td>
<td>is not applicable when running WLST in interactive mode.</td>
<td></td>
</tr>
<tr>
<td>isAdminServer</td>
<td>Boolean value specifying whether WLST is connected to a WebLogic Administration</td>
<td>wls:/mydomain/serverConfig&gt; print isAdminServer true</td>
</tr>
<tr>
<td></td>
<td>Server instance. WLST sets this variable to true if WLST is connected to a</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WebLogic Administration Server; otherwise, WLST sets it to false.</td>
<td></td>
</tr>
<tr>
<td>mbs</td>
<td>MBeanServerConnection object that corresponds to the current location in the</td>
<td>wls:/mydomain/serverConfig&gt; mbs.isRegistered(ObjectName('mydomain:Name=mydomain,Type=Domain'))</td>
</tr>
<tr>
<td></td>
<td>hierarchy.</td>
<td></td>
</tr>
<tr>
<td>recording</td>
<td>Boolean value specifying whether WLST is recording commands. WLST sets this</td>
<td>wls:/mydomain/serverConfig&gt; print recording true</td>
</tr>
<tr>
<td></td>
<td>variable to true when the startRecording command is entered; otherwise, WLST</td>
<td></td>
</tr>
<tr>
<td></td>
<td>sets this variable to false.</td>
<td></td>
</tr>
<tr>
<td>runtimeService</td>
<td>RuntimeServiceMBean MBean.</td>
<td>wls:/mydomain/serverConfig&gt; sr=runtimeService.getServerRuntime()</td>
</tr>
<tr>
<td>serverName</td>
<td>Name of the server to which WLST is connected.</td>
<td>wls:/mydomain/serverConfig&gt; print serverName myserver</td>
</tr>
<tr>
<td>typeService</td>
<td>TypeServiceMBean MBean.</td>
<td>wls:/mydomain/serverConfig&gt; mi=typeService.getMBeanInfo('weblogic.management.configuration.ServerMBean')</td>
</tr>
<tr>
<td>username</td>
<td>Name of user currently connected to WLST.</td>
<td>wls:/mydomain/serverConfig&gt; print username weblogic</td>
</tr>
<tr>
<td>version</td>
<td>Current version of the running server to which WLST is connected.</td>
<td>wls:/mydomain/serverConfig&gt; print version WebLogic Server 9.0 Thu Aug 31 12:15:50 PST 2005 778899</td>
</tr>
</tbody>
</table>
4

Infrastructure Security Custom WLST Commands

The following sections describe the Oracle Fusion Middleware Infrastructure Security custom WLST commands in detail. Topics include:

- Section 4.1, “Overview of WSLT Security Commands”
- Section 4.2, “Audit Configuration Commands”
- Section 4.3, “SSL Configuration Commands”
- Section 4.4, “Oracle Identity Federation Commands”
- Section 4.5, “Directory Integration Platform Commands”
- Section 4.6, “Security Commands”
- Section 4.7, “Oracle Access Manager Commands”

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

---

**Note:** To use the Infrastructure Security custom WLST commands, you must invoke the WLST script from the Oracle Common home. See "Using Custom WLST Commands" in the Oracle Fusion Middleware Administrator’s Guide.

---

4.1 Overview of WSLT Security Commands

WLST security commands are divided into the following categories:

| Table 4-1  WLST Command Categories |
|-----------------|----------------------------------|
| Command Category| Description                        |
| Audit Configuration Commands | View and manage audit policies and the audit repository configuration |
| SSL Configuration Commands | View and manage wallets, JKS keystores, and SSL configuration for Oracle HTTP Server, Oracle WebCache, Oracle Internet Directory, and Oracle Virtual Directory components. |
| Oracle Identity Federation Commands | View and manage configuration for Oracle Identity Federation |
| Directory Integration Platform Commands | For information on DIP tools, see "Directory Integration Platform Tools" in the Oracle Fusion Middleware User Reference for Oracle Identity Management |
4.2 Audit Configuration Commands

Use the WLST commands listed in Table 4–2 to view and manage audit policies and the audit repository configuration.

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>getNonJavaEEAuditMBeanName</td>
<td>Display the mBean name for a non-Java EE component.</td>
<td>Online</td>
</tr>
<tr>
<td>getAuditPolicy</td>
<td>Display audit policy settings.</td>
<td>Online</td>
</tr>
<tr>
<td>setAuditPolicy</td>
<td>Update audit policy settings.</td>
<td>Online</td>
</tr>
<tr>
<td>getAuditRepository</td>
<td>Display audit repository settings.</td>
<td>Online</td>
</tr>
<tr>
<td>setAuditRepository</td>
<td>Update audit repository settings.</td>
<td>Online</td>
</tr>
<tr>
<td>listAuditEvents</td>
<td>List audit events for one or all components.</td>
<td>Online</td>
</tr>
<tr>
<td>exportAuditConfig</td>
<td>Export a component’s audit configuration.</td>
<td>Online</td>
</tr>
<tr>
<td>importAuditConfig</td>
<td>Import a component’s audit configuration.</td>
<td>Online</td>
</tr>
</tbody>
</table>

For more information, see the Oracle Fusion Middleware Security Guide.

4.2.1 getNonJavaEEAuditMBeanName

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

4.2.1.1 Description

This command displays the mbean name for non-Java EE components given the instance name, component name, component type, and the name of the Oracle WebLogic Server on which the component’s audit mbean is running. The mbean name is a required parameter to other audit WLST commands when managing a non-Java EE component.

4.2.1.2 Syntax

getNonJavaEEAuditMBeanName(instName, compName, compType, svrName)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>instName</td>
<td>Specifies the name of the application server instance.</td>
</tr>
<tr>
<td>compName</td>
<td>Specifies the name of the component instance.</td>
</tr>
<tr>
<td>compType</td>
<td>Specifies the type of component. Valid values are ohs, oid, ovd, and WebCache.</td>
</tr>
<tr>
<td>svrName</td>
<td>Specifies the name of the Oracle WebLogic Server.</td>
</tr>
</tbody>
</table>
4.2.1.3 Example
The following interactive command displays the mBean name for an Oracle Internet Directory:

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

4.2.2 getAuditPolicy
Online command that displays the audit policy settings.

4.2.2.1 Description
This command displays audit policy settings including the filter preset, special users, custom events, maximum log file size, and maximum log directory size. The component mbean name is required for non-Java EE components like Oracle Internet Directory and Oracle Virtual Directory.

---

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

---

4.2.2.2 Syntax
```
getAuditPolicy([mbeanName])
```

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

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

```
wls:/mydomain/serverConfig> getAuditPolicy()
Location changed to domainRuntime tree. This is a read-only tree with DomainMBean as the root.
For more help, use help(domainRuntime)

FilterPreset:All
Max Log File Size:104857600
Max Log Dir Size:0
```

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

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

4.2.3 setAuditPolicy
Online command that updates an audit policy.

4.2.3.1 Description
Online command that configures the audit policy settings. You can set the filter preset, add or remove users, and add or remove custom events. The component mbean name
Audit Configuration Commands

is required for non-Java EE components like Oracle Internet Directory and Oracle Virtual Directory.

---

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

---

### 4.2.3.2 Syntax

```java
setAuditPolicy([mbeanName],[filterPreset],[addSpecialUsers],
[removeSpecialUsers],[addCustomEvents],[removeCustomEvents])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>mbeanName</code></td>
<td>Specifies the name of the component audit MBean for non-Java EE components.</td>
</tr>
<tr>
<td><code>filterPreset</code></td>
<td>Specifies the filter preset to be changed.</td>
</tr>
<tr>
<td><code>addSpecialUsers</code></td>
<td>Specifies the special users to be added.</td>
</tr>
<tr>
<td><code>removeSpecialUsers</code></td>
<td>Specifies the special users to be removed.</td>
</tr>
<tr>
<td><code>addCustomEvents</code></td>
<td>Specifies the custom events to be added.</td>
</tr>
<tr>
<td><code>removeCustomEvents</code></td>
<td>Specifies the custom events to be removed.</td>
</tr>
</tbody>
</table>

---

### 4.2.3.3 Examples

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

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

```
wls:/mydomain/serverConfig> getAuditPolicy();
Already in Domain Runtime Tree
FilterPreset:None
Special Users:user2,user3
Max Log File Size:104857600
Max Log Dir Size:0
```

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

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

```
wls:/mydomain/serverConfig> getAuditPolicy();
Already in Domain Runtime Tree
FilterPreset:Custom
Special Users:user2
Max Log File Size:104857600
Max Log Dir Size:0
```

The following interactive command sets audit policy to a `Low` level:

```
wls:/IDMDomain/domainRuntime> setAuditPolicy(filterPreset='Low');
Already in Domain Runtime Tree
Audit Policy Information updated successfully
```

```
wls:/IDMDomain/domainRuntime> getAuditPolicy();
Already in Domain Runtime Tree
FilterPreset:Low
Max Log File Size:104857600
Max Log Dir Size:0
```
The following command sets a custom filter to audit the CheckAuthorization event:

```
setAuditPolicy(filterPreset='Custom',
addCustomEvents='JPS:CheckAuthorization');
```

Audit Policy Information updated successfully

```
getAuditPolicy();
```

FilterPreset:Custom
Special Users:user1
Max Log File Size:104857600
Max Log Dir Size:0
Custom Events:JPS:CheckAuthorization

### 4.2.4 getAuditRepository

Online command that displays audit repository settings.

#### 4.2.4.1 Description

This command displays audit repository settings for Java EE components and applications (for other components like Oracle Internet Directory, the repository configuration resides in opmn.xml). Also displays database configuration if the repository is a database type.

#### 4.2.4.2 Syntax

```
getAuditRepository
```

#### 4.2.4.3 Example

The following command displays audit repository configuration:

```
getAuditRepository()
```

Repository Type:File

### 4.2.5 setAuditRepository

Online command that updates audit repository settings.

#### 4.2.5.1 Description

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

#### 4.2.5.2 Syntax

```
setAuditRepository([switchToDB],[dataSourceName],[interval])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>switchToDB</td>
<td>If true, switches the repository from file to database.</td>
</tr>
</tbody>
</table>
**4.2.5.3 Examples**

The following command switches from a file repository to a database repository:

```
setAuditRepository(switchToDB='true');
```

Audit Repository Information updated

```
getAuditRepository();
```

JNDI Name:jdbc/AuditDB
Interval:15
Repository Type:DB

The following interactive command changes audit repository to a specific database and sets the audit loader interval to 14 seconds:

```
setAuditRepository(switchToDB='true',dataSourceName='jdbcAuditDB',interval='14')
```

4.2.6 listAuditEvents

Online command that displays a component's audit events.

**4.2.6.1 Description**

This command displays a component's audit events and attributes. For non-Java EE components, pass the component mbean name as a parameter. Java EE applications and services like Oracle Platform Security Services (OPSS) do not need the mbean parameter. Without a component type, all generic attributes applicable to all components are displayed.

---

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

---

**4.2.6.2 Syntax**

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

**Argument** | **Definition**
--- | ---
`mbeanName` | Specifies the name of the component MBean.
`componentType` | Specifies the component type.

**4.2.6.3 Examples**

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

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

Already in Domain Runtime Tree
Common Attributes
ComponentType
Type of the component. For MAS integrated SystemComponents this is the
componentType
InstanceId
Name of the MAS Instance, that this component belongs to
HostId
DNS hostname of originating host
HostNwaddr
IP or other network address of originating host
ModuleId
ID of the module that originated the message. Interpretation is unique within
Component ID.
ProcessId
ID of the process that originated the message

The following command displays audit events for Oracle HTTP Server:

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

The following command displays all audit events:

wls:/IDMDomain/domainRuntime> listAuditEvents();
Already in Domain Runtime Tree

Components:
DIP
JPS
OIF
OWSM-AGENT
OWSM-PM-EJB
ReportsServer
WS-PolicyAttachment
WebCache
WebServices
Attributes applicable to all components:
ComponentType
InstanceId
HostId
HostNwaddr
ModuleId
ProcessId
OracleHome
HomeInstance
ECID
RID
...

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

4.2.7.1 Description
This command exports the audit configuration to a file. For non-Java EE components,
pass the component mbean name as a parameter. Java EE applications and services
like Oracle Platform Security Services (OPSS) do not need the mbean parameter.
4.2.7.2 Syntax

`exportAuditConfig([mbeanName],fileName)`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>mbeanName</td>
<td>Specifies the name of the non-Java EE component MBean.</td>
</tr>
<tr>
<td>fileName</td>
<td>Specifies the path and file name to which the audit configuration should be exported.</td>
</tr>
</tbody>
</table>

4.2.7.3 Examples

The following interactive command exports the audit configuration for a component:

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

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

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

4.2.8 importAuditConfig

Online command that imports a component's audit configuration.

4.2.8.1 Description

This command imports the audit configuration from an external file. For non-Java EE components, pass the component mbean name as a parameter. Java EE applications and services like Oracle Platform Security Services (OPSS) do not need the mbean parameter.

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

4.2.8.2 Syntax

`importAuditConfig([mbeanName],fileName)`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>mbeanName</td>
<td>Specifies the name of the non-Java EE component MBean.</td>
</tr>
<tr>
<td>fileName</td>
<td>Specifies the path and file name from which the audit configuration should be imported.</td>
</tr>
</tbody>
</table>

4.2.8.3 Examples

The following interactive command imports the audit configuration for a component:

```
ws:/mydomain/serverConfig>
importAuditConfig(on='oracle.security.audit.test:type=CSAuditMBean,name=CSAuditProxyMBean',fileName='/tmp/auditconfig')
```
The following interactive command imports the audit configuration for a component; no mBean is specified:

```
wlst:/mydomain/serverConfig> importAuditConfig(fileName='/tmp/auditconfig')
```

### 4.3 SSL Configuration Commands

Use the WLST commands listed in Table 4–3 to view and manage SSL configuration for Oracle Fusion Middleware components.

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>addCertificateRequest</td>
<td>Generate a certificate signing request in an Oracle wallet.</td>
<td>Online</td>
</tr>
<tr>
<td>addSelfSignedCertificate</td>
<td>Add a self-signed certificate to an Oracle wallet.</td>
<td>Online</td>
</tr>
<tr>
<td>changeKeyStorePassword</td>
<td>Change the password to a JKS keystore.</td>
<td>Online</td>
</tr>
<tr>
<td>changeWalletPassword</td>
<td>Change the password to an Oracle wallet.</td>
<td>Online</td>
</tr>
<tr>
<td>configureSSL</td>
<td>Set the SSL attributes for a component listener.</td>
<td>Online</td>
</tr>
<tr>
<td>createKeyStore</td>
<td>Create a JKS keystore.</td>
<td>Online</td>
</tr>
<tr>
<td>createWallet</td>
<td>Create an Oracle wallet.</td>
<td>Online</td>
</tr>
<tr>
<td>deleteKeyStore</td>
<td>Delete a JKS keystore.</td>
<td>Online</td>
</tr>
<tr>
<td>deleteWallet</td>
<td>Delete an Oracle wallet.</td>
<td>Online</td>
</tr>
<tr>
<td>exportKeyStore</td>
<td>Export a JKS keystore to a file.</td>
<td>Online</td>
</tr>
<tr>
<td>exportKeyStoreObject</td>
<td>Export an object from a JKS keystore to a file.</td>
<td>Online</td>
</tr>
<tr>
<td>exportWallet</td>
<td>Export an Oracle wallet to a file.</td>
<td>Online</td>
</tr>
<tr>
<td>exportWalletObject</td>
<td>Export an object from an Oracle wallet to a file.</td>
<td>Online</td>
</tr>
<tr>
<td>generateKey</td>
<td>Generate a keypair in a JKS keystore.</td>
<td>Online</td>
</tr>
<tr>
<td>getKeyStoreObject</td>
<td>Display a certificate or other object present in a JKS keystore.</td>
<td>Online</td>
</tr>
<tr>
<td>getSSL</td>
<td>Display the SSL attributes for a component listener.</td>
<td>Online</td>
</tr>
<tr>
<td>getWalletObject</td>
<td>Display a certificate or other object present in an Oracle wallet.</td>
<td>Online</td>
</tr>
<tr>
<td>importKeyStore</td>
<td>Import a JKS keystore from a file.</td>
<td>Online</td>
</tr>
<tr>
<td>importKeyStoreObject</td>
<td>Import a certificate or other object from a file to a JKS keystore.</td>
<td>Online</td>
</tr>
<tr>
<td>importWallet</td>
<td>Import an Oracle wallet from a file.</td>
<td>Online</td>
</tr>
<tr>
<td>importWalletObject</td>
<td>Import a certificate or other object from a file to an Oracle wallet.</td>
<td>Online</td>
</tr>
<tr>
<td>listKeyStoreObjects</td>
<td>List all objects present in a JKS keystore.</td>
<td>Online</td>
</tr>
<tr>
<td>listKeyStores</td>
<td>List all JKS keystores configured for a component instance.</td>
<td>Online</td>
</tr>
<tr>
<td>listWalletObjects</td>
<td>List all objects present in an Oracle wallet.</td>
<td>Online</td>
</tr>
<tr>
<td>listWallets</td>
<td>List all Oracle wallets configured for a component instance.</td>
<td>Online</td>
</tr>
</tbody>
</table>
4.3.1 addCertificateRequest

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

4.3.1.1 Description

This command generates a certificate signing request in Base64 encoded PKCS#10 format in an Oracle wallet for a component instance (Oracle HTTP Server, Oracle WebCache or Oracle Internet Directory). To get a certificate signed by a certificate authority (CA), send the certificate signing request to your CA.

4.3.1.2 Syntax

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

4.3.1.3 Example

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

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

4.3.2 addSelfSignedCertificate

Online command that adds a self-signed certificate.
4.3.2.1 Description
This command creates a key pair and wraps it in a self-signed certificate in an Oracle wallet for the specified component instance (Oracle HTTP Server, Oracle WebCache or Oracle Internet Directory). Only keys based on the RSA algorithm are generated.

4.3.2.2 Syntax
addSelfSignedCertificate(instName, compName, compType, walletName, password, DN, keySize)

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

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

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

4.3.3 changeKeyStorePassword
Online command that changes the keystore password.

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

4.3.3.2 Syntax
changeKeyStorePassword(instName, compName, compType, keystoreName, currPassword, newPassword)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>instName</td>
<td>Specifies the name of the application server instance.</td>
</tr>
<tr>
<td>compName</td>
<td>Specifies the name of the component instance.</td>
</tr>
<tr>
<td>compType</td>
<td>Specifies the type of component. Valid value is 'ovd'.</td>
</tr>
<tr>
<td>keystoreName</td>
<td>Specifies the filename of the keystore.</td>
</tr>
<tr>
<td>currPassword</td>
<td>Specifies the current keystore password.</td>
</tr>
<tr>
<td>newPassword</td>
<td>Specifies the new keystore password.</td>
</tr>
</tbody>
</table>
4.3.3 Example
The following command changes the password of file `keys.jks` for Oracle Virtual Directory instance ovdl in application server instance inst1:

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

4.3.4 changeWalletPassword
Online command that changes the password of an Oracle wallet.

4.3.4.1 Description
This command changes the password of an Oracle wallet for the specified component instance (Oracle HTTP Server, Oracle WebCache or Oracle Internet Directory). This command is only applicable to password-protected wallets.

4.3.4.2 Syntax
```
changeWalletPassword(instName, compName, compType, walletName, currPassword, newPassword)
```

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

4.3.4.3 Example
The following command changes the password for wallet1 from `currpassword` to `newpassword` for Oracle HTTP Server instance ohs1 in application server instance inst1:

```
<wls:/mydomain/serverConfig> changeWalletPassword('inst1', 'ohs1', 'ohs','wallet1', 'currpassword', 'newpassword')
```

4.3.5 configureSSL
Online command that sets SSL attributes.

4.3.5.1 Description
This command sets the SSL attributes for a component listener. The attributes are specified in a properties file format (name=value). If a properties file is not provided, or it does not contain any SSL attributes, default attribute values are used. For component-specific SSL attribute value defaults, see the chapter “SSL Configuration in Oracle Fusion Middleware” in the Oracle Fusion Middleware Administrator’s Guide.

4.3.5.2 Syntax
```
configureSSL(instName, compName, compType, listener, filePath)
```
4.3.5.3 Examples

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

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

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

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

4.3.6 createKeyStore

Online command that creates a JKS keystore.

4.3.6.1 Description

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

4.3.6.2 Syntax

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

### Argument Definition

**instName**
Specifies the name of the application server instance.

**compName**
Specifies the name of the component instance.

**compType**
Specifies the type of component. Valid values are 'oid', 'ovd', 'ohs', and 'webcache'.

**listener**
Specifies the name of the component listener to be configured for SSL.

**filePath**
Specifies the absolute path of the properties file containing the SSL attributes to set.

**keystoreName**
Specifies the filename of the keystore file to be created.

**password**
Specifies the keystore password.

4.3.6.3 Example

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

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

Online command that creates an Oracle wallet.

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

4.3.7.2 Syntax
createWallet(instName, compName, compType, walletName, password)

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

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

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

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

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

4.3.8 deleteKeyStore

Online command that deletes a keystore.

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

4.3.8.2 Syntax
deleteKeyStore(instName, compName, compType, keystoreName)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>instName</td>
<td>Specifies the name of the application server instance.</td>
</tr>
<tr>
<td>compName</td>
<td>Specifies the name of the component instance.</td>
</tr>
<tr>
<td>compType</td>
<td>Specifies the type of component. Valid value is 'ovd'.</td>
</tr>
<tr>
<td>keystoreName</td>
<td>Specifies the name of the keystore file to delete.</td>
</tr>
</tbody>
</table>
4.3.8.3 Example
The following command deletes JKS file keys.jks for Oracle Virtual Directory instance ovd1 in application server instance inst1:

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

4.3.9 deleteWallet
Online command that deletes an Oracle wallet.

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

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

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

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

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

4.3.10 exportKeyStore
Online command that exports the keystore to a file.

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

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>instName</td>
<td>Specifies the name of the application server instance.</td>
</tr>
<tr>
<td>compName</td>
<td>Specifies the name of the component instance.</td>
</tr>
<tr>
<td>compType</td>
<td>Specifies the type of component. Valid value is 'ovd'.</td>
</tr>
<tr>
<td>keystoreName</td>
<td>Specifies the name of the keystore file.</td>
</tr>
<tr>
<td>password</td>
<td>Specifies the password of the keystore.</td>
</tr>
</tbody>
</table>
4.3.10.3 Example

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

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

4.3.11 `exportKeyStoreObject`

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

4.3.11.1 Description

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

4.3.11.2 Syntax

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

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

4.3.11.3 Examples

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

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

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

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

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

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

### 4.3.12 exportWallet

Online command that exports an Oracle wallet.

#### 4.3.12.1 Description

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

#### 4.3.12.2 Syntax

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

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

#### 4.3.12.3 Examples

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

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

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

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

### 4.3.13 exportWalletObject

Online command that exports a certificate or other wallet object to a file.
4.3.13.1 Description
This command exports a certificate signing request, certificate, certificate chain or trusted certificate present in an Oracle wallet to a file for the specified component instance (Oracle HTTP Server, Oracle WebCache or Oracle Internet Directory). DN is used to indicate the object to be exported.

4.3.13.2 Syntax
exportWalletObject(instName, compName, compType, walletName, password, type, path, DN)

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

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

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

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

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

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

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

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

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

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

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

4.3.14.2 Syntax

\[
generateKey(instName, compName, compType, keystoreName, password, DN, keySize, alias, algorithm)
\]

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

4.3.14.3 Examples

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

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

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

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

4.3.15 getKeyStoreObject

Online command that shows details about a keystore object.

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

4.3.15.2 Syntax

\[
getKeyStoreObject(instName, compName, compType, keystoreName, password, type, \ldots)
\]
4.3.15.3 Examples
The following command shows a trusted certificate with index 1 present in keys.jks, for Oracle Virtual Directory instance ovd1, in application server instance inst1:

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

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

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

4.3.16 getSSL
Online command that lists the configured SSL attributes.

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

4.3.16.2 Syntax
getSSL(instName, compName, compType, listener)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>instName</td>
<td>Specifies the name of the application server instance.</td>
</tr>
<tr>
<td>compName</td>
<td>Specifies the name of the component instance.</td>
</tr>
<tr>
<td>compType</td>
<td>Specifies the type of component. Valid values are 'ovd', 'oid', 'ohs', and 'webcache'.</td>
</tr>
<tr>
<td>listener</td>
<td>Specifies the name of the component listener.</td>
</tr>
</tbody>
</table>

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

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

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

4.3.17.1 **Description**

This command displays a specific certificate signing request, certificate or trusted certificate present in an Oracle wallet for the specified component instance (Oracle HTTP Server, Oracle WebCache or Oracle Internet Directory). The wallet object is indicated by its index number, as given by the `listWalletObjects` command. For certificates or trusted certificates, it shows the certificate details including DN, key size, algorithm and other data. For certificate signing requests, it shows the subject DN, key size and algorithm.

4.3.17.2 **Syntax**

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

4.3.17.3 **Examples**

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

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

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

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

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

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

Online command that imports a keystore from a file.

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

4.3.18.2 Syntax
importKeyStore(instName, compName, compType, keystoreName, password, filePath)

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

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

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

4.3.19 importKeyStoreObject

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

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

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

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

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

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

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

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

4.3.20 importWallet

Online command that imports an Oracle wallet from a file.

4.3.20.1 Description

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

4.3.20.2 Syntax

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

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

4.3.20.3 Examples

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

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

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

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

### 4.3.21 importWalletObject

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

#### 4.3.21.1 Description

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

#### 4.3.21.2 Syntax

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

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

#### 4.3.21.3 Examples

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

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

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

```bash
wls:/mydomain/serverConfig> importWalletObject('inst1', 'oid1', 'oid', 'wallet1', 'password', 'Certificate', '/tmp/cert.txt')
```
The following command imports a trusted certificate from file trust.txt into wallet1, for Oracle Internet Directory instance oid1, in application server instance inst1:

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

### 4.3.22 listKeyStoreObjects

Online command that lists the contents of a keystore.

#### 4.3.22.1 Description

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

#### 4.3.22.2 Syntax

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

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

#### 4.3.22.3 Examples

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

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

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

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

### 4.3.23 listKeyStores

Online command that lists all the keystores for a component.

#### 4.3.23.1 Description

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

#### 4.3.23.2 Syntax

```
listKeyStores(instName, compName, compType)
```
4.3.23.3 Example
The following command lists all keystores for Oracle Virtual Directory instance *ovd1* in application server instance *inst1*:

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

### 4.3.24 listWalletObjects

Online command that lists all objects in an Oracle wallet.

#### 4.3.24.1 Description

This command lists all certificate signing requests, certificates, or trusted certificates present in an Oracle wallet for the specified component instance (Oracle HTTP Server, Oracle WebCache or Oracle Internet Directory).

#### 4.3.24.2 Syntax

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

#### Argument Definition

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

#### 4.3.24.3 Examples

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

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

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

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

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

```bash
wls:\mydomain\serverConfig> listWalletObjects('inst1', 'oid1', 'oid', 'wallet1', 'password', 'TrustedCertificate')
```
4.3.25 listWallets

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

4.3.25.1 Description

This command displays all the wallets configured for the specified component instance (Oracle HTTP Server, Oracle WebCache or Oracle Internet Directory), and identifies the auto-login wallets.

4.3.25.2 Syntax

listWallets(instName, compName, compType)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>instName</td>
<td>Specifies the name of the application server instance.</td>
</tr>
<tr>
<td>compName</td>
<td>Specifies the name of the component instance.</td>
</tr>
<tr>
<td>compType</td>
<td>Specifies the type of component. Valid values are 'ohs', 'oid', and 'webcache'.</td>
</tr>
</tbody>
</table>

4.3.25.3 Example

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

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

4.3.26 removeKeyStoreObject

Online command that removes an object from a keystore.

4.3.26.1 Description

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

4.3.26.2 Syntax

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>instName</td>
<td>Specifies the name of the application server instance.</td>
</tr>
<tr>
<td>compName</td>
<td>Specifies the name of the component instance.</td>
</tr>
<tr>
<td>compType</td>
<td>Specifies the type of component. Valid value is 'ovd'.</td>
</tr>
<tr>
<td>keystoreName</td>
<td>Specifies the name of the keystore file.</td>
</tr>
<tr>
<td>password</td>
<td>Specifies the password of the keystore.</td>
</tr>
<tr>
<td>type</td>
<td>Specifies the type of the keystore object to be removed. Valid values are 'Certificate', 'TrustedCertificate' or 'TrustedAll'.</td>
</tr>
</tbody>
</table>
### 4.3.26.3 Examples

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

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

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

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

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

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

### 4.3.27 removeWalletObject

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

#### 4.3.27.1 Description

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

#### 4.3.27.2 Syntax

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>instName</td>
<td>Specifies the name of the application server instance.</td>
</tr>
<tr>
<td>compName</td>
<td>Specifies the name of the component instance.</td>
</tr>
<tr>
<td>compType</td>
<td>Specifies the type of component. Valid values are 'ohs', 'oid', and 'webcache'.</td>
</tr>
<tr>
<td>walletName</td>
<td>Specifies the name of the wallet file.</td>
</tr>
<tr>
<td>password</td>
<td>Specifies the password of the wallet.</td>
</tr>
<tr>
<td>type</td>
<td>Specifies the type of the keystore object to be removed. Valid values are 'CertificateRequest', 'Certificate', 'TrustedCertificate' or 'TrustedAll'.</td>
</tr>
<tr>
<td>DN</td>
<td>Specifies the Distinguished Name of the wallet object to be removed.</td>
</tr>
</tbody>
</table>
4.3.27.3 Examples
The following command removes all trusted certificates from wallet1, for Oracle Internet Directory instance oid1, in application server instance inst1. It is not necessary to provide a DN, so we pass null (denoted by None) for the DN parameter:

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

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

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

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

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

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

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

4.4 Oracle Identity Federation Commands
Use the WLST commands listed in Table 4–4 to view and manage configuration for Oracle Identity Federation.

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>addConfigListEntryInMap</td>
<td>Add a configuration list entry to a map.</td>
<td>Online</td>
</tr>
<tr>
<td>addConfigMapEntryInMap</td>
<td>Add a configuration map entry to a map.</td>
<td>Online</td>
</tr>
<tr>
<td>addConfigPropertyListEntry</td>
<td>Add a configuration property list entry.</td>
<td>Online</td>
</tr>
<tr>
<td>addConfigPropertyMapEntry</td>
<td>Add a configuration property map entry to the map.</td>
<td>Online</td>
</tr>
<tr>
<td>addCustomAuthnEngine</td>
<td>Add a custom authentication engine.</td>
<td>Online</td>
</tr>
<tr>
<td>addCustomSPEngine</td>
<td>Add a custom SP engine.</td>
<td>Online</td>
</tr>
<tr>
<td>addFederationListEntryInMap</td>
<td>Add a federations list entry to the map.</td>
<td>Online</td>
</tr>
<tr>
<td>addFederationMapEntryInMap</td>
<td>Add a federation map entry to the map.</td>
<td>Online</td>
</tr>
<tr>
<td>addFederationPropertyListEntry</td>
<td>Add a federation property list entry.</td>
<td>Online</td>
</tr>
<tr>
<td>addFederationPropertyMapEntry</td>
<td>Add a federation property map entry.</td>
<td>Online</td>
</tr>
<tr>
<td>deleteCustomAuthnEngine</td>
<td>Delete a custom authentication engine.</td>
<td>Online</td>
</tr>
<tr>
<td>deleteCustomSPEngine</td>
<td>Delete a custom SP engine.</td>
<td>Online</td>
</tr>
<tr>
<td>deleteProviderFederation</td>
<td>Delete a provider from the federation.</td>
<td>Online</td>
</tr>
<tr>
<td>Use this command...</td>
<td>To...</td>
<td>Use with WLST...</td>
</tr>
<tr>
<td>-------------------------------------------------------</td>
<td>------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>deleteUserFederation</td>
<td>Delete a user from the federation.</td>
<td>Online</td>
</tr>
<tr>
<td>changeMessageStore</td>
<td>Change the message store to memory or RDBMS.</td>
<td>Online</td>
</tr>
<tr>
<td>changePeerProviderDescription</td>
<td>Change a peer provider’s description.</td>
<td>Online</td>
</tr>
<tr>
<td>changeSessionStore</td>
<td>Change the session store to memory or RDBMS.</td>
<td>Online</td>
</tr>
<tr>
<td>createConfigPropertyList</td>
<td>Create a configuration property list.</td>
<td>Online</td>
</tr>
<tr>
<td>createConfigPropertyListInMap</td>
<td>Create a configuration property list in the map.</td>
<td>Online</td>
</tr>
<tr>
<td>createConfigPropertyMap</td>
<td>Create a configuration property map.</td>
<td>Online</td>
</tr>
<tr>
<td>createConfigPropertyMapInMap</td>
<td>Create a nested configuration property map in a map.</td>
<td>Online</td>
</tr>
<tr>
<td>createFederationPropertyList</td>
<td>Create a federation property list.</td>
<td>Online</td>
</tr>
<tr>
<td>createFederationPropertyListInMap</td>
<td>Create a federation property list in the map.</td>
<td>Online</td>
</tr>
<tr>
<td>createFederationPropertyMap</td>
<td>Create a federation property map.</td>
<td>Online</td>
</tr>
<tr>
<td>createFederationPropertyMapInMap</td>
<td>Create a nested federation property map in a map.</td>
<td>Online</td>
</tr>
<tr>
<td>createPeerProviderEntry</td>
<td>Create a peer provider entry.</td>
<td>Online</td>
</tr>
<tr>
<td>getConfigListValueInMap</td>
<td>Retrieve a configuration list value from the map.</td>
<td>Online</td>
</tr>
<tr>
<td>getConfigMapEntryInMap</td>
<td>Retrieve a configuration map value from the map.</td>
<td>Online</td>
</tr>
<tr>
<td>getConfigProperty</td>
<td>Retrieve a configuration property entry.</td>
<td>Online</td>
</tr>
<tr>
<td>getConfigPropertyList</td>
<td>Retrieve a configuration property list.</td>
<td>Online</td>
</tr>
<tr>
<td>getConfigPropertyMapEntry</td>
<td>Retrieve a configuration property map entry.</td>
<td>Online</td>
</tr>
<tr>
<td>getFederationListValueInMap</td>
<td>Retrieve a federation list value from the map.</td>
<td>Online</td>
</tr>
<tr>
<td>getFederationMapEntryInMap</td>
<td>Retrieve a federation map entry from a nested map.</td>
<td>Online</td>
</tr>
<tr>
<td>getFederationProperty</td>
<td>Retrieve a federation property.</td>
<td>Online</td>
</tr>
<tr>
<td>getFederationPropertyList</td>
<td>Retrieve the federation property list.</td>
<td>Online</td>
</tr>
<tr>
<td>getFederationPropertyMapEntry</td>
<td>Retrieve a federation property map entry.</td>
<td>Online</td>
</tr>
<tr>
<td>listCustomAuthnEngines</td>
<td>Display the list of custom authentication engines.</td>
<td>Online</td>
</tr>
<tr>
<td>listCustomSPEngines</td>
<td>Display the list of custom SP engines.</td>
<td>Online</td>
</tr>
<tr>
<td>loadMetadata</td>
<td>Load metadata from a file.</td>
<td>Online</td>
</tr>
<tr>
<td>oifStatus</td>
<td>Display the current status of Oracle Identity Federation on the managed server.</td>
<td>Online</td>
</tr>
<tr>
<td>removeConfigListInMap</td>
<td>Delete a configuration list in the map.</td>
<td>Online</td>
</tr>
</tbody>
</table>
For more information, see the *Oracle Fusion Middleware Administrator’s Guide for Oracle Identity Federation*.

### 4.4.1 addConfigListEntryInMap

Online command that adds a property value to a map.

#### 4.4.1.1 Description

This command adds a property value to a nested list inside a map in config.xml.

#### 4.4.1.2 Syntax

```
addConfigListEntryInMap(configName, mapname, listName, value, type)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>configName</td>
<td>Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20..) to be updated.</td>
</tr>
<tr>
<td>mapname</td>
<td>Specifies the name of the property to map to be changed in config.xml.</td>
</tr>
<tr>
<td>listname</td>
<td>Specifies the name of the list.</td>
</tr>
<tr>
<td>value</td>
<td>Specifies the property value.</td>
</tr>
</tbody>
</table>
4.4.1.3 Example
The following command adds valueA to a map list in server configuration:

```plaintext
wls:/mydomain/serverConfig>
addConfigListEntryInMap('serverconfig','mymap','mylistA','valueA','string')
```

4.4.2 addConfigMapEntryInMap
Online command that adds a nested map property entry in a map.

4.4.2.1 Description
This command that adds a property name/value pair to a map nested inside a map in config.xml.

4.4.2.2 Syntax
```
addConfigMapEntryInMap(configName, mapname, nestedMapName, propName, value, type)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>configName</td>
<td>Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20,...) to be updated.</td>
</tr>
<tr>
<td>mapname</td>
<td>Specifies the name of the property map to be changed in config.xml.</td>
</tr>
<tr>
<td>nestedMapName</td>
<td>name of the nested property map to be changed.</td>
</tr>
<tr>
<td>propName</td>
<td>Specifies the name of the list.</td>
</tr>
<tr>
<td>value</td>
<td>Specifies the property value.</td>
</tr>
<tr>
<td>type</td>
<td>Specifies the type of property, BOOLEAN or STRING or LONG.</td>
</tr>
</tbody>
</table>

4.4.2.3 Example
The following command adds a boolean name/value pair to nestedmapB inside the map mymap.

```plaintext
wls:/mydomain/serverConfig>
addConfigMapEntryInMap('serverconfig','mymap','nestedmapB','myvarB','true','boolean')
```

4.4.3 addConfigPropertyListEntry
Online command that adds a list property entry to config.xml.

4.4.3.1 Description
This command adds a property value to a list in config.xml.

4.4.3.2 Syntax
```
addConfigPropertyListEntry(configName, listName, value, type)
```
4.4.3.3 Example
The following command adds a string value to mylistA.

```
wls:/mydomain/serverConfig>
addConfigPropertyListEntry('serverconfig','mylistA','valueA','string')
```

4.4.4 addConfigPropertyMapEntry
Online command that adds a property name/value entry in a map in config.xml.

4.4.4.1 Description
This command adds a property name/value entry in a map in config.xml.

4.4.4.2 Syntax
```
addConfigPropertyMapEntry(configName, mapName, propName, value, type)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>configName</td>
<td>Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20,..) to be updated.</td>
</tr>
<tr>
<td>mapName</td>
<td>Specifies the name of the property map to be added in config.xml.</td>
</tr>
<tr>
<td>propName</td>
<td>Specifies the name of the property map.</td>
</tr>
<tr>
<td>value</td>
<td>Specifies the property map value to be added.</td>
</tr>
<tr>
<td>type</td>
<td>Specifies the type of property, BOOLEAN or STRING or LONG.</td>
</tr>
</tbody>
</table>

4.4.4.3 Example
The following command adds valueA of string type to a map.

```
wls:/mydomain/serverConfig>
addConfigPropertyMapEntry('serverconfig','mymapA','myvarA','valueA','string')
```

4.4.5 addCustomAuthnEngine
Online command that adds a custom authentication integration engine.

4.4.5.1 Description
This command adds a custom authentication integration engine to config.xml.

4.4.5.2 Syntax
```
addCustomAuthnEngine(name, [enabled], [webContext], [authnRelativePath], 
[logoutRelativePath], [logoutEnabled])
```

Argument Definition
- `configname`: Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20,..) to be updated.
- `mapname`: Specifies the name of the property map to be added in config.xml.
- `listname`: Specifies the new property list value. The entered value is appended to the list.
- `type`: Specifies the type of property, BOOLEAN or STRING or LONG.
4.4.5.3 Example
The following command defines an engine named test and enables it.

```
wls:/mydomain/serverConfig> addCustomAuthnEngine('test','true')
```

4.4.6 addCustomSPEngine
Online command that adds a custom service provider (SP) engine.

4.4.6.1 Description
This command adds a custom SP integration engine to config.xml.

4.4.6.2 Syntax
```
addCustomSPEngine(name, [enabled, [authnMech], [webContext], [authnRelativePath],
[logoutRelativePath], [logoutEnabled])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Specifies the name of the custom engine.</td>
</tr>
<tr>
<td>enabled</td>
<td>This flag specifies whether the engine is enabled (true) or not (false).</td>
</tr>
<tr>
<td>authnMech</td>
<td>Specifies the authentication mechanism for the engine.</td>
</tr>
<tr>
<td>webContext</td>
<td>Specifies the web context for the engine.</td>
</tr>
<tr>
<td>authnRelativePath</td>
<td>Specifies the authentication relative path URL for the engine.</td>
</tr>
<tr>
<td>logoutRelativePath</td>
<td>Specifies the logout relative path URL for the engine.</td>
</tr>
<tr>
<td>logoutEnabled</td>
<td>This flag is set true to enable logout for the engine, else false.</td>
</tr>
</tbody>
</table>

4.4.6.3 Example
The following command adds an engine and gives it a disabled status.

```
addCustomSPEngine('new engine','false','oracle:fed:authentication:unspecified','webcontext')
```

4.4.7 addFederationListEntryInMap
Online command that adds a list property entry in a map.

4.4.7.1 Description
This command adds a property value to a nested list inside a map in cot.xml.
4.4.7.2 Syntax
addFederationListEntryInMap(providerID, mapname, listName, value, type)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>providerID</td>
<td>Specifies the provider ID.</td>
</tr>
<tr>
<td>mapname</td>
<td>Specifies the name of the property map to be changed in cot.xml.</td>
</tr>
<tr>
<td>listName</td>
<td>Specifies the name of the property list to be added to the map.</td>
</tr>
<tr>
<td>value</td>
<td>Specifies the property list value to be added. The entered value is appended to the list.</td>
</tr>
<tr>
<td>type</td>
<td>Specifies the type of property, BOOLEAN or STRING or LONG.</td>
</tr>
</tbody>
</table>

4.4.7.3 Example
The following command adds a boolean property list to mymap.

```
wls:/mydomain/serverConfig>
addFederationListEntryInMap('providerB','mymap','mylistB','true','boolean')
```

4.4.8 addFederationMapEntryInMap
Online command that adds a nested map property entry in a map.

4.4.8.1 Description
This command adds a property name/value pair to a map nested inside a map in cot.xml.

4.4.8.2 Syntax
addFederationMapEntryInMap(providerID, mapname, nestedMapName, propName, value, type)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>providerID</td>
<td>Specifies the provider ID.</td>
</tr>
<tr>
<td>mapname</td>
<td>Specifies the name of the property map to be changed in cot.xml.</td>
</tr>
<tr>
<td>nestedMapName</td>
<td>Specifies the name of the nested property map to be changed.</td>
</tr>
<tr>
<td>propName</td>
<td>Specifies the name of the property to be updated in the map.</td>
</tr>
<tr>
<td>value</td>
<td>Specifies the property value to be added. The entered value is appended to the list.</td>
</tr>
<tr>
<td>type</td>
<td>Specifies the type of property, BOOLEAN or STRING or LONG.</td>
</tr>
</tbody>
</table>

4.4.8.3 Example
The following command adds a value of type string to the myvarA property in a nested map.

```
wls:/mydomain/serverConfig>
addFederationMapEntryInMap('providerA','mymap','nestedmapA','myvarA','valueA','string')
```

4.4.9 addFederationPropertyListEntry
Online command that adds a list property entry.
4.4.9.1 Description
This command adds a property value to a list in cot.xml.

4.4.9.2 Syntax
addFederationPropertyListEntry(providerID, listName, value, type)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>providerID</td>
<td>Specifies the provider ID.</td>
</tr>
<tr>
<td>listName</td>
<td>Specifies the name of the property list to be updated.</td>
</tr>
<tr>
<td>value</td>
<td>Specifies the property list value to be added. The entered value is appended to the list.</td>
</tr>
<tr>
<td>type</td>
<td>Specifies the type of property, BOOLEAN or STRING or LONG.</td>
</tr>
</tbody>
</table>

4.4.9.3 Example
The following command adds a value in string format to a specified property list.

wls:/mydomain/serverConfig>
addFederationPropertyListEntry('providerA','mylistA','valueA','string')

4.4.10 addFederationPropertyMapEntry
Online command that a property name/value entry in a map.

4.4.10.1 Description
This command adds a property name/value pair to a map in cot.xml.

4.4.10.2 Syntax
addFederationPropertyMapEntry(providerID, mapName, propName, value, type)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>providerID</td>
<td>Specifies the provider ID.</td>
</tr>
<tr>
<td>mapName</td>
<td>Specifies the name of the property map to be changed in cot.xml.</td>
</tr>
<tr>
<td>propName</td>
<td>Specifies the name of the property to be added in the map.</td>
</tr>
<tr>
<td>value</td>
<td>Specifies the property value to be added. The entered value is appended to the list.</td>
</tr>
<tr>
<td>type</td>
<td>Specifies the type of property, BOOLEAN or STRING or LONG.</td>
</tr>
</tbody>
</table>

4.4.10.3 Example
The following command adds boolean property myvarB to a map.

wls:/mydomain/serverConfig>
addFederationPropertyMapEntry('providerA','mymapB','myvarB','true','boolean')

4.4.11 deleteCustomAuthnEngine
Online command that deletes a custom authentication integration engine from the configuration.
4.4.11.1 Description
This command deletes a custom authentication integration engine in config.xml. You must provide the engine ID for an existing custom authentication engine in config.xml.

4.4.11.2 Syntax
deleteCustomAuthnEngine(engineID)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>engineID</td>
<td>Specifies the engine ID of an existing engine to be deleted.</td>
</tr>
</tbody>
</table>

4.4.11.3 Example
The following command deletes the authentication engine with ID id1234.

```
wls:/mydomain/serverConfig> deleteCustomAuthnEngine ('id1234')
```

4.4.12 deleteCustomSPEngine
Online command that deletes a custom service provider (SP) integration engine from the configuration.

4.4.12.1 Description
This command deletes a custom SP integration engine in config.xml. The EngineID for an existing custom SP engine in config.xml must be provided.

4.4.12.2 Syntax
ddeleteCustomSPEngine(engineID)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>engineID</td>
<td>Specifies the engine ID of an existing engine to be deleted.</td>
</tr>
</tbody>
</table>

4.4.12.3 Example
The following command deletes the engine with ID id1234.

```
wls:/mydomain/serverConfig> deleteCustomSPEngine ('id1234')
```

4.4.13 deleteProviderFederation
Online command that deletes federations for given provider.

4.4.13.1 Description
This command deletes federations for given provider ID.

4.4.13.2 Syntax
deleteProviderFederation(providerID)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>providerID</td>
<td>Specifies the ProviderID for the peer provider for which federation is to be deleted.</td>
</tr>
</tbody>
</table>
4.4.13.3 Example
The following command deletes providerA:

```
wlst/mydomain/serverConfig> deleteProviderFederation(providerA)
```

4.4.14 deleteUserFederation
Online command that deletes federations for given users.

4.4.14.1 Description
This command deletes federations for the given list of users.

4.4.14.2 Syntax
```
deleteUserFederation([user1,...])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>user1</td>
<td>Specifies a comma-separated list of users whose federations are to be deleted. At least one user must be specified.</td>
</tr>
</tbody>
</table>

4.4.14.3 Example
The following command deletes federations for three users:

```
wlst/mydomain/serverConfig> deleteUserFederation(['userA','userB','userC'])
```

4.4.15 changeMessageStore
Online command that changes the message store between memory and RDBMS.

4.4.15.1 Description
This command changes the message store to memory or RDBMS.

4.4.15.2 Syntax
```
changeMessageStore(type, [jndiname])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Specifies the type of store, RDBMS or Memory. Default is Memory.</td>
</tr>
<tr>
<td>jndiname</td>
<td>Specifies the jndi name to set for the store. Required if type is RDBMS.</td>
</tr>
</tbody>
</table>

4.4.15.3 Example
The following command changes the message store to RDBMS:

```
wlst/mydomain/serverConfig> changeMessageStore('RDBMS','jdbc/mydb')
```

4.4.16 changePeerProviderDescription
Online command that changes the peer provider description.

4.4.16.1 Description
This command updates a peer provider description in cot.xml.
4.4.16.2 Syntax
changePeerProviderDescription(providerID, description)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>providerID</td>
<td>Specifies the provider ID.</td>
</tr>
<tr>
<td>description</td>
<td>Specifies the provider description.</td>
</tr>
</tbody>
</table>

4.4.16.3 Example
The following command updates the description of a provider:
```
wlis:/mydomain/serverConfig> changePeerProviderDescription('providerA','new description')
```

4.4.17 changeSessionStore
Online command that changes the session store between memory and RDBMS.

4.4.17.1 Description
This command changes the session store to memory or RDBMS.

4.4.17.2 Syntax
changeSessionStore(type, [jndiname])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Specifies the type of store, RDBMS or Memory. Default is Memory.</td>
</tr>
<tr>
<td>jndiname</td>
<td>Specifies the jndi name to set for the store. Required if type is RDBMS.</td>
</tr>
</tbody>
</table>

4.4.17.3 Example
The following command changes the session store to RDBMS.
```
wls:/mydomain/serverConfig> changeSessionStore('RDBMS','jdbc/mydb')
```

4.4.18 createConfigPropertyList
Online command that creates a property list.

4.4.18.1 Description
This command creates a property list in config.xml.

4.4.18.2 Syntax
createConfigPropertyList(configName, listName)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>configName</td>
<td>Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20,..) to be updated.</td>
</tr>
<tr>
<td>listName</td>
<td>Specifies the property list name.</td>
</tr>
</tbody>
</table>

4.4.18.3 Example
The following command creates property list mylistA.
4.4.19 createConfigPropertyListInMap

Online command that creates a property list nested in the property map.

4.4.19.1 Description
This command creates a property list, nested in the property map, in config.xml.

4.4.19.2 Syntax
createConfigPropertyListInMap(configName, mapName, listName)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>configName</td>
<td>Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20,..) to be updated.</td>
</tr>
<tr>
<td>mapName</td>
<td>Specifies an existing property map to contain the nested list.</td>
</tr>
<tr>
<td>listName</td>
<td>Specifies the property list name.</td>
</tr>
</tbody>
</table>

4.4.19.3 Example
The following command creates property list mylistA nested in a property map.

wls:/mydomain/serverConfig>
createConfigPropertyListInMap('serverconfig','mymapA','mylistA')

4.4.20 createConfigPropertyMap

Online command that creates a property map.

4.4.20.1 Description
This command that creates a property map in config.xml.

4.4.20.2 Syntax
createConfigPropertyMap(configName, mapName)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>configName</td>
<td>Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20,..) to be updated.</td>
</tr>
<tr>
<td>mapName</td>
<td>Specifies the property map to create.</td>
</tr>
</tbody>
</table>

4.4.20.3 Example
The following command creates property map mymapA:

wls:/mydomain/serverConfig> createConfigPropertyMap('serverconfig','mymapA')

4.4.21 createConfigPropertyMapInMap

Online command that creates a property map.

4.4.21.1 Description
This command that creates a property map in config.xml.
4.4.21.2 Syntax
createConfigPropertyMapInMap(configName, mapName, nestedMapName)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>configName</td>
<td>Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20..) to be updated.</td>
</tr>
<tr>
<td>mapName</td>
<td>Specifies the name of an existing property map.</td>
</tr>
<tr>
<td>nestedMapName</td>
<td>Specifies the name of the property map to create nested inside mapName.</td>
</tr>
</tbody>
</table>

4.4.21.3 Example
The following command creates nested property map nestedmymapA:

wls:/mydomain/serverConfig>
createConfigPropertyMapInMap('serverconfig', 'mymapA', 'nestedmapA')

4.4.22 createFederationPropertyList
Online command that creates a property list.

4.4.22.1 Description
This command creates a property list in cot.xml.

4.4.22.2 Syntax
createFederationPropertyList(providerID, listName)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>providerID</td>
<td>Specifies the provider ID.</td>
</tr>
<tr>
<td>listName</td>
<td>Specifies the name of the property list.</td>
</tr>
</tbody>
</table>

4.4.22.3 Example
The following command creates property list mylistA:

wls:/mydomain/serverConfig> createFederationPropertyList('providerA', 'mylistA')

4.4.23 createFederationPropertyListInMap
Online command that creates a property list nested in a property map.

4.4.23.1 Description
This command creates a property list, nested in a property map, in cot.xml.

4.4.23.2 Syntax
createFederationPropertyListInMap(providerID, mapName, listName)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>providerID</td>
<td>Specifies the provider ID.</td>
</tr>
<tr>
<td>mapName</td>
<td>Specifies an existing property map to contain the nested list.</td>
</tr>
<tr>
<td>listName</td>
<td>Specifies the name of the property list.</td>
</tr>
</tbody>
</table>
4.4.23.3 Example
The following command creates nested property list mylistA:

```bash
wls:/mydomain/serverConfig>
createFederationPropertyListInMap('providerA', 'mymapA', 'mylistA')
```

4.4.24 createFederationPropertyMap
Online command that creates a property map.

4.4.24.1 Description
This command that creates a property map in cot.xml.

4.4.24.2 Syntax
```
createFederationPropertyMap(providerID, mapName)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>providerID</td>
<td>Specifies the provider ID.</td>
</tr>
<tr>
<td>mapName</td>
<td>Specifies the name of the property map to be added to cot.xml.</td>
</tr>
</tbody>
</table>

4.4.24.3 Example
The following command creates property map mymapA:

```bash
wls:/mydomain/serverConfig> createFederationPropertyMap('providerA', 'mymapA')
```

4.4.25 createFederationPropertyMapInMap
Online command that creates a nested property map.

4.4.25.1 Description
This command that creates a property map, nested in another property map, in cot.xml.

4.4.25.2 Syntax
```
createFederationPropertyMapInMap(providerID, mapName, nestedMapName)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>providerID</td>
<td>Specifies the provider ID.</td>
</tr>
<tr>
<td>mapName</td>
<td>Specifies the name of an existing property map.</td>
</tr>
<tr>
<td>nestedMapName</td>
<td>Specifies the name of the property map to be nested inside mapName in cot.xml.</td>
</tr>
</tbody>
</table>

4.4.25.3 Example
The following command creates nested property map nestedmapA:

```bash
wls:/mydomain/serverConfig>
createFederationPropertyMapInMap('providerA', 'mymapA', 'nestedmapA')
```

4.4.26 createPeerProviderEntry
Online command that creates a peer provider property map entry.
4.4.26.1 Description
This command creates a peer provider as a Map property entry to cot.xml.

4.4.26.2 Syntax
createPeerProviderEntry(providerID, description, providerType, version)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>providerID</td>
<td>Specifies the provider ID to be created.</td>
</tr>
<tr>
<td>description</td>
<td>This is the description of the provider ID.</td>
</tr>
<tr>
<td>providerType</td>
<td>Specifies the provider type of the peer provider to be created.</td>
</tr>
<tr>
<td>version</td>
<td>Specifies the version of the peer provider to be created.</td>
</tr>
</tbody>
</table>

4.4.26.3 Example
The following command creates a SAML 2.0 service provider:

wls:/mydomain/serverConfig> createPeerProviderEntry('providerA','idp
test','SP','SAML2.0')

4.4.27 getConfigListValueInMap
Online command that returns a list nested in a map.

4.4.27.1 Description
This command returns a list, nested in a map, from config.xml.

4.4.27.2 Syntax
getConfigListValueInMap(configName, mapName, listName)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>configName</td>
<td>Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20,..) to be accessed.</td>
</tr>
<tr>
<td>mapName</td>
<td>Specifies the name of the property map.</td>
</tr>
<tr>
<td>listName</td>
<td>Specifies the name of the list to be fetched from the map.</td>
</tr>
</tbody>
</table>

4.4.27.3 Example
The following command returns mylistA:

wls:/mydomain/serverConfig>
getConfigListValueInMap('serverConfig','mymapA','mylistA')

4.4.28 getConfigMapEntryInMap
Online command that returns a map property entry nested in a map.

4.4.28.1 Description
This command returns a map property entry, nested in a map, from config.xml.

4.4.28.2 Syntax
getConfigMapEntryInMap(configName, mapName, nestedMapName, propName)
4.4.28.3 Example
The following command returns property entry myvarA:

```
wls:/mydomain/serverConfig>
getConfigMapEntryInMap('serverconfig','mymap','nestedmapA','myvarA')
```

4.4.29 getConfigProperty
Online command that returns a property value.

4.4.29.1 Description
This command returns a property value from config.xml.

4.4.29.2 Syntax
```
getConfigProperty(configName, propName)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>configName</td>
<td>Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20,..) to be accessed.</td>
</tr>
<tr>
<td>propName</td>
<td>Specifies the name of the property to be fetched from the nested map.</td>
</tr>
</tbody>
</table>

4.4.29.3 Example
The following command returns property myvarA:

```
wls:/mydomain/serverConfig> getConfigProperty('serverconfig','myvarA')
```

4.4.30 getConfigPropertyList
Online command that returns a property list.

4.4.30.1 Description
This command returns a property list from config.xml.

4.4.30.2 Syntax
```
getConfigPropertyList(configName, listName)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>configName</td>
<td>Specifies the configuration name.</td>
</tr>
<tr>
<td>listName</td>
<td>Specifies the name of the property list to be fetched from config.xml.</td>
</tr>
</tbody>
</table>
4.4.30.3 Example
The following command returns mylistA:

```
wlst:/mydomain/serverConfig> getConfigPropertyList('serverconfig','mylistA')
```

4.4.31 getConfigPropertyMapEntry
Online command that returns a property value from a map.

4.4.31.1 Description
This command returns a property value from a map in config.xml.

4.4.31.2 Syntax
```
getConfigPropertyMapEntry(configName, mapName, propName)
```

4.4.31.3 Example
The following command returns property propA:

```
wls:/mydomain/serverConfig> getConfigPropertyMapEntry('serverconfig','mapA', 'propA')
```

4.4.32 getFederationListValueInMap
Online command that returns a list value nested in a map.

4.4.32.1 Description
This command returns a list value nested in a map from cot.xml.

4.4.32.2 Syntax
```
getFederationListValueInMap(providerID, mapName, listName)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>providerID</td>
<td>Specifies the name of the peer provider to be accessed.</td>
</tr>
<tr>
<td>mapName</td>
<td>Specifies the name of the property map.</td>
</tr>
<tr>
<td>listName</td>
<td>Specifies the name of the list to be fetched from the map.</td>
</tr>
</tbody>
</table>

4.4.32.3 Example
The following command returns nested list mylistA:

```
wls:/mydomain/serverConfig> getFederationListValueInMap('providerA','mymapA','mylistA')
```
4.4.33 getFederationMapEntryInMap

Online command that returns a map property entry nested in a map.

4.4.33.1 Description
This command returns a map property entry, nested in a map, from cot.xml.

4.4.33.2 Syntax
getFederationMapEntryInMap(providerID, mapname, nestedMapName, propName)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>providerID</td>
<td>Specifies the name of the peer provider to be accessed.</td>
</tr>
<tr>
<td>mapName</td>
<td>Specifies the name of the property map.</td>
</tr>
<tr>
<td>nestedMapName</td>
<td>Specifies the name of the nested property map.</td>
</tr>
<tr>
<td>propName</td>
<td>Specifies the name of the property to be fetched from the nested map.</td>
</tr>
</tbody>
</table>

4.4.33.3 Example
The following command returns property entry myvarA:

```
wls:/mydomain/serverConfig>
getFederationMapEntryInMap('providerA','mymap','nestedmapA','myvarA')
```

4.4.34 getFederationProperty

Online command that returns a property value.

4.4.34.1 Description
This command returns a property value from cot.xml.

4.4.34.2 Syntax
getFederationProperty(providerID, propName)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>providerID</td>
<td>Specifies the name of the peer provider to be accessed.</td>
</tr>
<tr>
<td>propName</td>
<td>Specifies the name of the property to be fetched from cot.xml.</td>
</tr>
</tbody>
</table>

4.4.34.3 Example
The following command returns property myvarA:

```
wls:/mydomain/serverConfig> getFederationProperty('providerA','myvarA')
```

4.4.35 getFederationPropertyList

Online command that returns a property list.

4.4.35.1 Description
This command returns a property list from cot.xml.
4.4.35.2 Syntax
getFederationPropertyList(providerID, listName)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>providerID</td>
<td>Specifies the name of the peer provider to be accessed.</td>
</tr>
<tr>
<td>listName</td>
<td>Specifies the name of the list to be fetched from the map.</td>
</tr>
</tbody>
</table>

4.4.35.3 Example
The following command returns list mylistA:

```
wls:/mydomain/serverConfig> getFederationPropertyList('providerA','mylistA')
```

4.4.36 getFederationPropertyMapEntry
Online command that returns a property value from a map.

4.4.36.1 Description
This command returns a property value from a map in cot.xml.

4.4.36.2 Syntax
getFederationPropertyMapEntry(providerID, mapName, propName)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>providerID</td>
<td>Specifies the name of the peer provider to be accessed.</td>
</tr>
<tr>
<td>mapName</td>
<td>Specifies the name of the property map.</td>
</tr>
<tr>
<td>propName</td>
<td>Specifies the name of the property to be fetched from the nested map.</td>
</tr>
</tbody>
</table>

4.4.36.3 Example
The following command returns property propA from a map:

```
wls:/mydomain/serverConfig> getFederationPropertyMapEntry('providerA','mapA', 'propA')
```

4.4.37 listCustomAuthnEngines
Online command that returns a list of custom authentication integration engines.

4.4.37.1 Description
This command returns a list of custom authentication integration engines from config.xml.

4.4.37.2 Syntax
listCustomAuthnEngines()

4.4.37.3 Example
The following command returns the list of all SP engines:

```
wls:/mydomain/serverConfig> listCustomAuthnEngines()
```
4.4.38 listCustomSPEngines

Online command that returns a list of custom SP integration engines.

4.4.38.1 Description
This command returns a list of custom service provider (SP) integration engines from config.xml.

4.4.38.2 Syntax
listCustomSPEngines()

4.4.38.3 Example
The following command returns the list of all SP integration engines:

```
> wls:/mydomain/serverConfig> listCustomSPEngines()
```

4.4.39 loadMetadata

Online command that loads metadata from an input file.

4.4.39.1 Description
This command loads metadata from an input file into cot.xml.

4.4.39.2 Syntax
loadMetadata(metadatafile,description)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>metadatafile</td>
<td>Specifies the metadata file of the peer provider to be added or updated.</td>
</tr>
<tr>
<td>description</td>
<td>This is a brief description of the peer provider to be loaded.</td>
</tr>
</tbody>
</table>

4.4.39.3 Example
The following command loads metadata from the file metadatafile.xml:

```
> wls:/mydomain/serverConfig> loadMetadata('/home/metadatafile.xml','some description')
```

4.4.40 oifStatus

Online command that reports the current status of the Oracle Identity Federation application in the managed server to which WLST is connected.

4.4.40.1 Description
This command displays the current status of Oracle Identity Federation on the managed server.

4.4.40.2 Syntax
loifStatus('serverurl', 'configfile', 'keyfile')

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>serverurl</td>
<td>Specifies the URL of the managed server.</td>
</tr>
</tbody>
</table>
4.4.40.3 Example
The following command provides no arguments; WLST prompts you for the Oracle WebLogic Server username, password, and the managed server URL, then displays the federation server status:

```bash
wls:/mydomain/serverConfig> oifStatus()
```

The following command provides only the managed server URL; WLST prompts you for the Oracle WebLogic Server username and password:

```bash
wls:/mydomain/serverConfig> oifStatus('', '', 't3://localhost:7499')
```

The following command provides all arguments needed for WLST to display the federation server status:

```bash
wls:/mydomain/serverConfig> oifStatus('configfileA', 'keyfileB', 't3://localhost:7499')
```

4.4.41 removeConfigListInMap
Online command that removes a list property nested in a map.

4.4.41.1 Description
This command removes a list property nested in a map from config.xml.

4.4.41.2 Syntax
```
removeConfigListInMap(configName, mapName, listName)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>configName</td>
<td>Specifies the name of the configuration (for example, idpsaml20, serverconfig, saml20...) to be accessed.</td>
</tr>
<tr>
<td>mapName</td>
<td>Specifies the name of the property map.</td>
</tr>
<tr>
<td>listName</td>
<td>Specifies the name of the list to be removed from the map.</td>
</tr>
</tbody>
</table>

4.4.41.3 Example
The following command removes the list property mylistA:

```bash
wls:/mydomain/serverConfig> removeConfigListInMap('serverConfig','mymapA','mylistA')
```

4.4.42 removeConfigMapEntryInMap
Online command that removes a map property nested in a map.

4.4.42.1 Description
This command removes a map property entry nested in a map from config.xml.
4.4.42.2 Syntax
```
removeConfigMapEntryInMap(configName, mapName, nestedMapName, propName)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>configName</td>
<td>Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20,..) to be accessed.</td>
</tr>
<tr>
<td>mapName</td>
<td>Specifies the name of the property map.</td>
</tr>
<tr>
<td>nestedMapName</td>
<td>Specifies the name of the nested property map.</td>
</tr>
<tr>
<td>propName</td>
<td>Specifies the name of the property to be removed from the nested map.</td>
</tr>
</tbody>
</table>

4.4.42.3 Example
The following command removes the nested property myvarA:

```
wls:/mydomain/serverConfig>
removeConfigMapEntryInMap('serverconfig','mymap','nestedmapA','myvarA')
```

4.4.43 removeConfigMapInMap

Online command that removes a map property nested in a map.

4.4.43.1 Description
This command removes a map property entry nested in a map from config.xml.

4.4.43.2 Syntax
```
removeConfigMapEntryInMap(configName, mapName, nestedMapName, propName)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>configName</td>
<td>Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20,..) to be updated.</td>
</tr>
<tr>
<td>mapName</td>
<td>Specifies the name of the property map.</td>
</tr>
<tr>
<td>nestedMapName</td>
<td>Specifies the name of the nested property map.</td>
</tr>
<tr>
<td>propName</td>
<td>Specifies the name of the property to be removed from the nested map.</td>
</tr>
</tbody>
</table>

4.4.43.3 Example
The following command removes the nested property myvarA:

```
wls:/mydomain/serverConfig>
removeConfigMapEntryInMap('serverconfig','mymap','nestedmapA','myvarA')
```

4.4.44 removeConfigProperty

Online command that removes a configuration property.

4.4.44.1 Description
This command removes a property from config.xml.

4.4.44.2 Syntax
```
removeConfigProperty(configName, propName)
```

4.4.44.3 Example
The following command removes the property myvarA:

```wls:/mydomain/serverConfig> removeConfigProperty('serverconfig','myvarA')```

4.4.45 removeConfigPropertyList
Online command that removes a configuration property list.

4.4.45.1 Description
This command removes a property list from config.xml.

4.4.45.2 Syntax
```
removeConfigPropertyList(configName, listName)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>configName</code></td>
<td>Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20,..) to be updated.</td>
</tr>
<tr>
<td><code>listName</code></td>
<td>Specifies the name of the property list to be removed.</td>
</tr>
</tbody>
</table>

4.4.45.3 Example
The following command removes the property list mylistA:

```wls:/mydomain/serverConfig> removeConfigPropertyList('serverconfig','mylistA')```

4.4.46 removeConfigPropertyMap
Online command that removes a property map.

4.4.46.1 Description
This command removes a property map in config.xml.

4.4.46.2 Syntax
```
removeConfigPropertyMap(configName, mapName)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>configName</code></td>
<td>Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20,..) to be updated.</td>
</tr>
<tr>
<td><code>mapName</code></td>
<td>Specifies the name of the property map to be removed.</td>
</tr>
</tbody>
</table>

4.4.46.3 Example
The following command removes mapA:

```wls:/mydomain/serverConfig> removeConfigPropertyMap('serverconfig','mapA')```
4.4.47 removeConfigPropertyMapEntry

Online command that removes a property value from a map.

4.4.47.1 Description
This command removes a property value from a map in config.xml.

4.4.47.2 Syntax
removeConfigPropertyMapEntry(configName, mapName, propName)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>configName</td>
<td>Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20,..) to be updated.</td>
</tr>
<tr>
<td>mapName</td>
<td>Specifies the name of the property map to be updated.</td>
</tr>
<tr>
<td>propName</td>
<td>Specifies the name of the property to be removed from the map.</td>
</tr>
</tbody>
</table>

4.4.47.3 Example
The following command removes property propA:

```
web:/mydomain/serverConfig> removeConfigPropertyMapEntry('serverconfig','mapA','propA')
```

4.4.48 removeFederationListInMap

Online command that removes a property list in a map.

4.4.48.1 Description
This command removes a property list in a map, in cot.xml.

4.4.48.2 Syntax
removeFederationListInMap(providerID, mapName, listName)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>providerID</td>
<td>Specifies the name of the peer provider to be accessed.</td>
</tr>
<tr>
<td>mapName</td>
<td>Specifies the name of the property map.</td>
</tr>
<tr>
<td>listName</td>
<td>Specifies the name of the property list to be removed.</td>
</tr>
</tbody>
</table>

4.4.48.3 Example
The following command removes mylistA in mymapA:

```
web:/mydomain/serverConfig> removeFederationListInMap('providerA','mymapA','mylistA')
```

4.4.49 removeFederationMapInMap

Online command that removes a nested map in a map.

4.4.49.1 Description
This command removes a property map nested inside a map in cot.xml.
4.4.49.2 Syntax

removeFederationMapInMap(providerID, mapname, nestedMapName)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>providerID</td>
<td>Specifies the name of the peer provider to be accessed.</td>
</tr>
<tr>
<td>mapName</td>
<td>Specifies the name of the property map containing the nested map.</td>
</tr>
<tr>
<td>nestedMapName</td>
<td>Specifies the name of the nested property map to be removed.</td>
</tr>
</tbody>
</table>

4.4.49.3 Example

The following command removes nestedmapA in mymap:

```
wls:/mydomain/serverConfig>
removeFederationMapInMap('providerA','mymap','nestedmapA')
```

4.4.50 removeFederationMapEntryInMap

Online command that removes a nested map property entry in a map.

4.4.50.1 Description

This command removes a property name/value pair to a map nested inside a map in cot.xml.

4.4.50.2 Syntax

removeFederationMapEntryInMap(providerID, mapname, nestedMapName, propName)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>providerID</td>
<td>Specifies the name of the peer provider to be accessed.</td>
</tr>
<tr>
<td>mapName</td>
<td>Specifies the name of the property map containing the nested map.</td>
</tr>
<tr>
<td>nestedMapName</td>
<td>Specifies the name of the nested property map.</td>
</tr>
<tr>
<td>propName</td>
<td>Specifies the name of the property to be removed from the nested map.</td>
</tr>
</tbody>
</table>

4.4.50.3 Example

The following command removes map property entry myvarA:

```
wls:/mydomain/serverConfig>
removeFederationMapEntryInMap('providerA', 'mymap', 'nestedmapA', 'myvarA')
```

4.4.51 removeFederationProperty

Online command that removes a property value.

4.4.51.1 Description

This command removes a property entry in cot.xml.

4.4.51.2 Syntax

removeFederationProperty(providerID, propName)
4.4.51.3 Example
The following command removes the provider property myvarA:

```
removeFederationProperty('providerA','myvarA')
```

4.4.52 removeFederationPropertyList
Online command that removes a property list entry.

4.4.52.1 Description
This command removes a property list entry in cot.xml.

4.4.52.2 Syntax
```
removeFederationPropertyList(providerID, listName)
```

4.4.52.3 Example
The following command removes mylistA:

```
removeFederationPropertyList('providerA','mylistA')
```

4.4.53 removeFederationPropertyMap
Online command that removes a property map.

4.4.53.1 Description
This command removes a property map in cot.xml.

4.4.53.2 Syntax
```
removeFederationPropertyMap(providerID, mapName)
```

4.4.53.3 Example
The following command removes a map:

```
removeFederationPropertyMap('providerA','mapA')
```
4.4.54 removeFederationPropertyMapEntry

Online command that removes a property value from a map.

4.4.54.1 Description
This command removes a property value from a map in cot.xml.

4.4.54.2 Syntax
removeFederationPropertyMapEntry(providerID, mapName, propName)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>providerID</td>
<td>Specifies the name of the peer provider to be accessed.</td>
</tr>
<tr>
<td>mapName</td>
<td>Specifies the name of the property map to be updated.</td>
</tr>
<tr>
<td>propName</td>
<td>Specifies the name of the property to be removed from the map.</td>
</tr>
</tbody>
</table>

4.4.54.3 Example
The following command removes property propA from a map:

```
wl:/mydomain/serverConfig> removeFederationPropertyMapEntry('providerA','mapA', 'propA')
```

4.4.55 removePeerProviderEntry

Online command that removes a peer provider entry.

4.4.55.1 Description
This command removes a peer provider entry from cot.xml.

4.4.55.2 Syntax
removePeerProviderEntry(providerID)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>providerID</td>
<td>Specifies the name of the peer provider to be removed.</td>
</tr>
</tbody>
</table>

4.4.55.3 Example
The following command removes providerA:

```
wls:/mydomain/serverConfig> removePeerProviderEntry('providerA')
```

4.4.56 setConfigProperty

Online command that sets a property value in config.xml.

4.4.56.1 Description
This command adds or updates a property value in config.xml.

4.4.56.2 Syntax
setConfigProperty(configname, propName, value, type)
Oracle Identity Federation Commands

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>configName</td>
<td>Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20...) to be updated.</td>
</tr>
<tr>
<td>propName</td>
<td>Specifies the name of the property to be added/updated in config.xml.</td>
</tr>
<tr>
<td>value</td>
<td>Specifies the property value.</td>
</tr>
<tr>
<td>type</td>
<td>Specifies the type of property, BOOLEAN or STRING or LONG.</td>
</tr>
</tbody>
</table>

### 4.4.56.3 Example
The following command sets the property myvarA and its value in the server configuration:

```
wls:/mydomain/serverConfig>
setConfigProperty('serverconfig','myvarA','myvalA','string')
```

### 4.4.57 setCustomAuthnEngine
Online command that updates a custom authentication integration engine.

#### 4.4.57.1 Description
This command updates a custom authentication integration engine in config.xml.

#### 4.4.57.2 Syntax

```
setCustomAuthnEngine(engineID, name, [enabled], [webContext], [authnRelativePath], [logoutRelativePath], [logoutEnabled])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>engineID</td>
<td>Specifies the engine ID of an existing engine.</td>
</tr>
<tr>
<td>name</td>
<td>Specifies the name of the custom engine.</td>
</tr>
<tr>
<td>enabled</td>
<td>This flag specifies whether the engine is enabled (true) or not (false).</td>
</tr>
<tr>
<td>webContext</td>
<td>Specifies the web context for the engine.</td>
</tr>
<tr>
<td>authnRelativePath</td>
<td>Specifies the authentication relative path URL for the engine.</td>
</tr>
<tr>
<td>logoutRelativePath</td>
<td>Specifies the logout relative path URL for the engine.</td>
</tr>
<tr>
<td>logoutEnabled</td>
<td>This flag is set true to enable logout for the engine, else false.</td>
</tr>
</tbody>
</table>

#### 4.4.57.3 Example
The following command updates the configuration of custom authentication engine abcdef:

```
wlsl/mydomain/serverConfig> setCustomAuthnEngine('abcdef',
'custom one','false','oracle:fed:authentication:unspecified','webcontext')
```

### 4.4.58 setCustomSPEngine
Online command that updates a custom SP integration engine.

#### 4.4.58.1 Description
This command updates an existing custom SP integration engine in config.xml.
4.4.58.2 Syntax

```plaintext
setCustomSPEngine(engineID, name, [enabled, [authnMech], [webContext], [authnRelativePath], [logoutRelativePath], [logoutEnabled])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>engineID</td>
<td>Specifies the engine ID of an existing custom engine.</td>
</tr>
<tr>
<td>name</td>
<td>Specifies the name of the custom engine.</td>
</tr>
<tr>
<td>enabled</td>
<td>This flag specifies whether the engine is enabled (true) or not (false).</td>
</tr>
<tr>
<td>authnMech</td>
<td>Specifies the authentication mechanism for the engine.</td>
</tr>
<tr>
<td>webContext</td>
<td>Specifies the web context for the engine.</td>
</tr>
<tr>
<td>authnRelativePath</td>
<td>Specifies the authentication relative path URL for the engine.</td>
</tr>
<tr>
<td>logoutRelativePath</td>
<td>Specifies the logout relative path URL for the engine.</td>
</tr>
<tr>
<td>logoutEnabled</td>
<td>This flag is set true to enable logout for the engine, else false.</td>
</tr>
</tbody>
</table>

4.4.58.3 Example

The following command sets the name and the enabled flag for the engine with ID `engineID2`:

```
wlsc:mydomain/serverConfig> setCustomSPEngine('engineID2', 'test', 'true')
```

4.4.59 setFederationProperty

Online command that adds or updates a property value.

4.4.59.1 Description

This command adds a property entry or updates an existing entry in cot.xml.

4.4.59.2 Syntax

```plaintext
setFederationProperty(providerID, propName, value, type)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>providerID</td>
<td>Specifies the name of the peer provider to be updated.</td>
</tr>
<tr>
<td>propName</td>
<td>Specifies the name of the property to be added/updated in cot.xml.</td>
</tr>
<tr>
<td>value</td>
<td>Specifies the property value.</td>
</tr>
<tr>
<td>type</td>
<td>Specifies the type of property, BOOLEAN or STRING or LONG.</td>
</tr>
</tbody>
</table>

4.4.59.3 Example

The following command creates the property `myvarA` and sets its value:

```
wls:mydomain/serverConfig> setFederationProperty('providerA', 'myvarA', 'myvalA', 'string')
```

4.5 Directory Integration Platform Commands

Some of the Directory Integration Platform (DIP) tools use WLST internally, and therefore, there are no custom WLST commands available to run from the WLST command prompt or to use within scripts. For information on DIP tools, see "Directory
4.6 Security Commands

Use the WLST security commands listed in Table 4–5 to operate on a domain policy or credential store, and to migrate policies and credentials from a source repository to a target repository.

Table 4–5 WLST Security Commands

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>createAppRole</td>
<td>Create a new application role.</td>
<td>Online</td>
</tr>
<tr>
<td>deleteAppRole</td>
<td>Remove an application role.</td>
<td>Online</td>
</tr>
<tr>
<td>grantAppRole</td>
<td>Add a principal to a role.</td>
<td>Online</td>
</tr>
<tr>
<td>revokeAppRole</td>
<td>Remove a principal from a role.</td>
<td>Online</td>
</tr>
<tr>
<td>listAppRoles</td>
<td>List all roles in an application.</td>
<td>Online</td>
</tr>
<tr>
<td>listAppRolesMembers</td>
<td>List all members in an application role.</td>
<td>Online</td>
</tr>
<tr>
<td>grantPermission</td>
<td>Create a new permission.</td>
<td>Online</td>
</tr>
<tr>
<td>revokePermission</td>
<td>Remove a permission.</td>
<td>Online</td>
</tr>
<tr>
<td>listPermissions</td>
<td>List all permissions granted to a principal.</td>
<td>Online</td>
</tr>
<tr>
<td>deleteAppPolicies</td>
<td>Remove all policies in an application.</td>
<td>Online</td>
</tr>
<tr>
<td>migrateSecurityStore</td>
<td>Migrate policies or credentials from a source repository to a target repository.</td>
<td>Offline</td>
</tr>
<tr>
<td>listCred</td>
<td>Obtain the list of attribute values of a credential.</td>
<td>Online</td>
</tr>
<tr>
<td>updateCred</td>
<td>Modify the attribute values of a credential.</td>
<td>Online</td>
</tr>
<tr>
<td>createCred</td>
<td>Create a new credential.</td>
<td>Online</td>
</tr>
<tr>
<td>deleteCred</td>
<td>Remove a credential.</td>
<td>Online</td>
</tr>
<tr>
<td>modifyBootStrapCredential</td>
<td>Update bootstrap credential store</td>
<td>Offline</td>
</tr>
<tr>
<td>reassociateSecurityStore</td>
<td>Reassociate policies and credentials to an LDAP repository</td>
<td>Online</td>
</tr>
<tr>
<td>upgradeSecurityStore</td>
<td>Upgrade security data from data used with release 10.1.x to data used with release 11.</td>
<td>Offline</td>
</tr>
</tbody>
</table>

4.6.1 createAppRole

Online command that creates a new application role.

4.6.1.1 Description

Creates a new application role in the domain policy store with a given application and role name. In the event of an error, the command returns a WLSTException.

4.6.1.2 Syntax

createAppRole(appStripe, appRoleName)
4.6.1.3 Example
The following invocation creates a new application role with application stripe myApp and role name myRole:

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

4.6.2 deleteAppRole
Online command that removes an application role.

4.6.2.1 Description
Removes an application role in the domain policy store with a given application and role name. In the event of an error, the command returns a WLSTException.

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

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

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

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

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

4.6.3.1 Description
Adds a principal (class or name) to a role with a given application stripe and name. In the event of an error, the command returns a WLSTException.

4.6.3.2 Syntax
```
grantAppRole(appStripe, appRoleName, principalClass, principalName)
```

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

Argument Definition

- **appStripe**: Specifies an application stripe.
- **appRoleName**: Specifies a role name.
4.6.3 Example
The following invocation adds a principal to the role with application stripe `myApp` and role name `myRole`:

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

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

4.6.4.1 Description
Removes a principal (class or name) from a role with a given application stripe and name. In the event of an error, the command returns a `WLSTException`.

4.6.4.2 Syntax
```
revokeAppRole(appStripe, appRoleName, principalClass, principalName)
```

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

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

4.6.5 listAppRoles
Online command that lists all roles in an application.

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

4.6.5.2 Syntax
```
listAppRoles(appStripe)
```

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

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

Online command that lists all members in a role.

### 4.6.6.1 Description

Lists all members in a role with a given application stripe and role name. In the event of an error, the command returns a WLSTException.

### 4.6.6.2 Syntax

```
listAppRoleMembers(appStripe, appRoleName)
```

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

### 4.6.6.3 Example

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

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

4.6.7 **grantPermission**

Online command that creates a new permission.

### 4.6.7.1 Description

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

### 4.6.7.2 Syntax

Optional arguments are enclosed in between square brackets.

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

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

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

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

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

4.6.8 revokePermission
Online command that removes a permission.

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

4.6.8.2 Syntax
Optional arguments are enclosed in between square brackets.

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

### Argument Definition

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

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

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

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

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

4.6.9 listPermissions

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

4.6.9.1 Description

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

4.6.9.2 Syntax

Optional arguments are enclosed in between square brackets.

listPermissions([appStripe,] principalClass, principalName)

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

4.6.9.3 Examples

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

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

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

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

4.6.10 deleteAppPolicies

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

4.6.10.1 Description

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

4.6.10.2 Syntax

delteAppPolicies(appStripe)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appStripe</td>
<td>Specifies an application stripe. If not specified, the command works on system policies.</td>
</tr>
</tbody>
</table>

4.6.10.3 Example

The following invocation removes all policies of application myApp:
4.6.11 migrateSecurityStore

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

4.6.11.1 Description

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

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

4.6.11.2 Syntax

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

Optional arguments are enclosed in square brackets.

To migrate identities, use the following syntax:

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

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

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

To migrate just system policies, use the following syntax:

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

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

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

To migrate all credentials, use the following syntax:

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

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

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

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

### 4.6.11.3 Examples

The following invocation illustrates the migration of the file-based policies of application `PolicyServlet1` to file-based policies of application `PolicyServlet2`:

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

The above invocation assumes that:

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

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

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

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

  ```
wls:/mydomain/serverConfig> migrateSecurityStore(type="credStore",
configFile="jps-config.xml", src="default1", dst="context2")
```

  The above invocation assumes that:

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

    ```xml
    <serviceInstance name="credstore1" provider="some.provider">
      <property name="location" value="/credstore1/cwallet.sso"/>
    </serviceInstance>
    <serviceInstance name="credstore2" provider="some.provider">
      <property name="location" value="/credstore2/cwallet.sso"/>
    </serviceInstance>
    ```
4.6.12 listCred

Online command that returns the list of attribute values of a credential in the domain credential store.

4.6.12.1 Description

Returns the list of attribute values of a credential in the domain credential store with given map name and key name. This command lists the data encapsulated in credentials of type password only. In the event of an error, the command returns a WLSTException.

4.6.12.2 Syntax

listCred(map, key)

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

4.6.12.3 Example

The following invocation returns all the information (such as user name, password, URL, port, and description) in the credential with map name myMap and key name myKey:

wls:/mydomain/serverConfig> listCred(map="myMap", key="myKey")

4.6.13 updateCred

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

4.6.13.1 Description

Modifies the type, user name, password, URL, and port number of a credential in the domain credential store with given map name and key name. This command can update the data encapsulated in credentials of type password only. In the event of an error, the command returns a WLSTException. This command runs in interactive mode only.

4.6.13.2 Syntax

Optional arguments are enclosed in square brackets.

updateCred(map, key, user, password, [desc])
4.6.13.3 Examples
The following invocation updates a password credential with the specified data:

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

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

4.6.14.1 Description
Creates a new credential in the domain credential store with a given map name, key name, type, user name and password, URL and port number. In the event of an error, the command returns a WLSTException. This command runs in interactive mode only.

4.6.14.2 Syntax
Optional arguments are enclosed in square brackets.

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

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

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

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

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

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

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

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

4.6.15.3 Examples

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

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

4.6.16 modifyBootStrapCredential

Offline command that updates a bootstrap credential store.

4.6.16.1 Description

Updates a bootstrap credential store with given user name and password. In the event of an error, the command returns a `WLSTException`.

Typically used in the following scenario: suppose that the domain policy and credential stores are LDAP-based, and the credentials to access the LDAP store (stored in the LDAP server) are changed. Then this command can be used to seed those changes into the bootstrap credential store.

4.6.16.2 Syntax

```
modifyBootStrapCredential(jpsConfigFile, username, password)
```

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

4.6.16.3 Examples

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

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

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

Any output regarding the audit service can be disregarded.

4.6.17 reassociateSecurityStore

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

4.6.17.2 Syntax
`reassociateSecurityStore(domain, admin, password, ldapurl, servertype, jpsroot [, join])`

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

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

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

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

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

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

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

To upgrade 10.1.x XML identity data to 11 XML identity data, use the following syntax:
updateSecurityStore(type="xmlIdStore", jpsConfigFile, srcJaznDataFile, srcRealm, dst)

To upgrade a 10.1.x XML policy data to 11 XML policy data, use the following syntax:
updateSecurityStore(type="xmlPolicyStore", jpsConfigFile, srcJaznDataFile, dst)

To upgrade a 10.1.x OID LDAP-based policy data to 11 XML policy data, use the following syntax:
updateSecurityStore(type="oidPolicyStore", jpsConfigFile, srcJaznDataFile, dst)

To upgrade a 10.1.x XML credential data to 11 XML credential data, use the following syntax:
updateSecurityStore(type="xmlCredStore", jpsConfigFile, srcJaznDataFile, users, dst)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Specifies the kind of security data being upgraded. The only valid values are xmlIdStore, xmlPolicyStore, oidPolicyStore, and xmlCredStore.</td>
</tr>
<tr>
<td>jpsConfigFile</td>
<td>Specifies the location of a configuration file jps-config.xml relative to the directory where the command is run. The target store of the upgrading is read from the context specified with the argument dst.</td>
</tr>
<tr>
<td>srcJaznDataFile</td>
<td>Specifies the location of a 10.1.x jazn data file relative to the directory where the command is run. This argument is required if the specified type is xmlIdStore, xmlPolicyStore, or xmlCredStore.</td>
</tr>
<tr>
<td>srcJaznConfigFile</td>
<td>Specifies the location of a 10.1.x jazn configuration file relative to the directory where the command is run. This argument is required if the specified type is oidPolicyStore.</td>
</tr>
<tr>
<td>srcRealm</td>
<td>Specifies the name of the realm from which identities need be migrated. This argument is required if the specified type is xmlIdStore.</td>
</tr>
<tr>
<td>users</td>
<td>Specifies a comma-separated list of users each formatted as realmName/userName. This argument is required if the specified type is xmlCredStore.</td>
</tr>
<tr>
<td>dst</td>
<td>Specifies the name of the jpsContext in the file passed to the argument jpsConfigFile where the destination store is configured. Optional. If unspecified, it defaults to the default context in the file passed in the argument jpsConfigFile.</td>
</tr>
</tbody>
</table>
4.6.18.3 Examples

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

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

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

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

4.7 Oracle Access Manager Commands

Use the WLST commands listed in Table 4–6 to manage Oracle Access Manager (OAM)-related components, such as authorization providers, identity asserters, and SSO providers.

<table>
<thead>
<tr>
<th>Table 4–6 WLST OAM Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use this command...</td>
</tr>
<tr>
<td>listOAMAuthnProviderParams</td>
</tr>
<tr>
<td>createOAMIdentityAsserter</td>
</tr>
<tr>
<td>updateOAMIdentityAsserter</td>
</tr>
<tr>
<td>createOAMAuthenticator</td>
</tr>
<tr>
<td>deleteOAMAuthnProvider</td>
</tr>
<tr>
<td>updateOAMAuthenticator</td>
</tr>
<tr>
<td>addOAMSSOProvider</td>
</tr>
</tbody>
</table>

4.7.1 listOAMAuthnProviderParams

Online command that lists the values of the parameters in effect in a domain OAM authenticator or identity asserter.

4.7.1.1 Description

Lists the values of the parameters set for a given OAM authenticator or identity asserter. In the event of an error, the command returns a WLSTException.

4.7.1.2 Syntax

```
listOAMAuthnProviderParams(name)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Specifies the name of the authenticator or identity asserter.</td>
</tr>
</tbody>
</table>
4.7.1.3 Example
The following invocation lists the parameters and values set for the asserter named myIdAsseter:

```
wls:/mydomain/serverConfig> listOAMAuthnProviderParams(name="myIdAsseter")
```

4.7.2 createOAMIdentityAsserter
Online command that creates an OAM identity asserter in the current domain.

4.7.2.1 Description
Creates an identity asserter with a given name in the current domain. Before executing this command, make sure that no OAM identity asserter is already configured in the current domain. In the event of an error, the command returns a WLSTException.

4.7.2.2 Syntax
```
createOAMIdentityAsserter(name)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Specifies the name of the new identity asserter. If no name is specified, it defaults to &quot;OAMIdentityAsserter&quot;.</td>
</tr>
</tbody>
</table>

4.7.2.3 Example
The following invocation creates a new identity asserter named OAMIdentityAsserter:
```
wls:/mydomain/serverConfig> createOAMIdentityAsserter()
```

4.7.3 updateOAMIdentityAsserter
Online command that updates the values of parameters of the OAM identity asserter in the current domain.

4.7.3.1 Description
Updates the value of given parameters of the domain OAM identity asserter. In the event of an error, the command returns a WLSTException.

4.7.3.2 Syntax
```
updateOAMIdentityAsserter(name, paramNameValueList)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Specifies the name of the OAM identity asserter whose parameter values to update.</td>
</tr>
</tbody>
</table>
4.7.3.3 Example

The following invocation updates the parameters accessGateName, accessGatePwd, pAccessServer, and ssoHeaderName in the OAM identity asserter named myIdAsserter:

```
wlst:/mydomain/serverConfig> updateOAMIdentity Asserter(name="myIdAsserter",
accessGateName='OAM IAP AccessGate', accessGatePwd='welcome1',
pAccessServer='myhost.domain.com:5543', ssoHeaderName='OAM_SSO_HEADER')
```

4.7.4 createOAMAuthenticator

Online command that creates an OAM authenticator in the current domain.

4.7.4.1 Description

Creates an OAM authenticator with a given name in the current domain. Before executing this command, make sure that no OAM authenticator is already configured in the default security domain. In the event of an error, the command returns a WLSTException.
4.7.4.2 Syntax
createOAMAuthenticator(name)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Specifies the name of the new authentication provider in the default domain. If no name is specified, it defaults to &quot;OAMAuthenticator&quot;.</td>
</tr>
</tbody>
</table>

4.7.4.3 Example
The following invocation creates a new authentication provider named OAMAuthenticator:

```
wlsp:mydomain/serverConfig> createOAMAuthenticator()
```

4.7.5 deleteOAMAuthnProvider
Online command that deletes the OAM authenticator from the current domain.

4.7.5.1 Description
Deletes the OAM authenticator with a given name from the current domain. In the event of an error, the command returns a WLSTException.

4.7.5.2 Syntax
deleteOAMAuthnProvider(name)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Specifies the name of the authentication provider to delete.</td>
</tr>
</tbody>
</table>

4.7.5.3 Example
The following invocation deletes the authenticator myAuthenticator:

```
wlsp:mydomain/serverConfig> deleteOAMAuthnProvider(name="myAuthenticator")
```

4.7.6 updateOAMAuthenticator
Online command that updates the values of parameters of the OAM authenticator in the current domain.

4.7.6.1 Description
Updates the value of given parameters of the domain OAM authenticator. In the event of an error, the command returns a WLSTException.

4.7.6.2 Syntax
updateOAMAuthenticator(name, paramNameValueList)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Specifies the name of the OAM authenticator whose parameter values to update.</td>
</tr>
</tbody>
</table>
4.7.6.3 Example
The following invocation updates the parameters accessGateName, accessGatePwd, and pAccessServer in the OAM authenticator named myAuthenticator:

```
$ wls:/mydomain/serverConfig> updateOAMAuthenticator(name="myAuthenticator", accessGateName='OAM AP AccessGate', accessGatePwd='welcome1', pAccessServer='myhost.domain.com:5543')
```

4.7.7 addOAMSSOProvider
Online command that adds an OAM SSO provider.

4.7.7.1 Description
Adds an SSO provider with given login URI, logout URI, and auto-login URI. This command modifies the domain jps-config.xml by adding an OAM SSO service.
instance with the required properties. In the event of an error, the command returns a WLSTException.

### 4.7.7.2 Syntax

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>loginuri</td>
<td>Specifies the URI of the login page. Required.</td>
</tr>
</tbody>
</table>
| logouturi    | Specifies the URI of the logout page. Optional. If unspecified, defaults to logouturi=NONE. 

Set to "" to ensure that ADF security calls the OPSS logout service, which uses the implementation of the class OAMSSOServiceImpl to clear the cookie ObSSOCookie.

More generally, an ADF-secured web application that would like to clear cookies without logging out the user should use this setting.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>autologinuri</td>
<td>Specifies the URI of the autologin page. Optional. If unspecified, it defaults to autologin=NONE.</td>
</tr>
</tbody>
</table>

### 4.7.7.3 Example

The following invocation adds an SSO provider with the passed URIs; note the special behavior implied by the setting logouturi="", as explained in the table above:

```bash
wls:/mydomain/serverConfig>
addOAMSSOProvider(loginuri="/${app.context}/adfAuthentication", logouturi="", autologin="/fooBar.cgi")
```
This chapter describes WebLogic Scripting Tool (WLST) commands for Oracle WebCenter. These commands enable you to configure WebCenter applications from the command-line. For additional details about WebCenter application configuration, see Oracle Fusion Middleware Administrator’s Guide for Oracle WebCenter.

Notes: 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 Oracle Fusion Middleware Administrator’s Guide.

Most configuration changes made using WebCenter WLST commands are only effective after you restart the Managed Server on which the WebCenter application is deployed. The only exceptions are the External Applications, Portlet Producers, and WebCenter Import and Export WLST commands.

WebCenter WLST commands are described in the following sections:

- Section 5.1, "Oracle WebCenter WLST Command Categories"
- Section 5.2, "General"
- Section 5.3, "Content Repository"
- Section 5.4, "Discussions and Announcements"
- Section 5.5, "External Applications"
- Section 5.6, "Instant Messaging and Presence"
- Section 5.7, "Mail"
- Section 5.8, "Personal Events"
- Section 5.9, "Portlet Producers"
- Section 5.10, "RSS"
- Section 5.11, "Search"
- Section 5.12, "WebCenter Spaces Crawlers"
- Section 5.13, "Worklists"
- Section 5.14, "WebCenter Spaces Workflows"
- Section 5.15, "Wikis and Blogs"
5.1 Oracle WebCenter WLST Command Categories

Oracle WebCenter WLST commands are grouped into the following categories (Table 5–1).

Most configuration changes made using WebCenter WLST commands are only effective after you restart the Managed Server on which the WebCenter application is deployed. The only exceptions are the External Applications, Portlet Producers, and WebCenter Import and Export WLST commands.

<table>
<thead>
<tr>
<th>Command Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
<td>Manage WebCenter connections.</td>
</tr>
<tr>
<td>Content Repository</td>
<td>Manage content repository connections and configure the Documents service.</td>
</tr>
<tr>
<td>Discussions and Announcements</td>
<td>Manage discussions server connections.</td>
</tr>
<tr>
<td>External Applications</td>
<td>Manage external application connections.</td>
</tr>
<tr>
<td>Instant Messaging and Presence</td>
<td>Manage instant messaging and presence server connections.</td>
</tr>
<tr>
<td>Mail</td>
<td>Manage mail server connections.</td>
</tr>
<tr>
<td>Personal Events</td>
<td>Manage personal event server connections.</td>
</tr>
<tr>
<td>Portlet Producers</td>
<td>Manage portlet producers.</td>
</tr>
<tr>
<td>RSS</td>
<td>Manage proxy settings for the RSS service.</td>
</tr>
<tr>
<td>Search</td>
<td>Manage Oracle Secure Enterprise Search (SES) connections and other search-related properties.</td>
</tr>
<tr>
<td>WebCenter Spaces Crawlers</td>
<td>Manage search crawlers for WebCenter Spaces.</td>
</tr>
<tr>
<td>Worklists</td>
<td>Manage BPEL server connections.</td>
</tr>
<tr>
<td>WebCenter Spaces Workflows</td>
<td>Manage the BPEL connection for WebCenter Spaces workflows.</td>
</tr>
<tr>
<td>Wikis and Blogs</td>
<td>Manage wiki and blog server connections.</td>
</tr>
<tr>
<td>WebCenter Identity Store</td>
<td>Configure options for searching a WebCenter application’s identity store.</td>
</tr>
<tr>
<td>WebCenter Import and Export</td>
<td>Export and import WebCenter Spaces applications, group spaces, group space templates, and producer metadata.</td>
</tr>
</tbody>
</table>

5.2 General

Use the General commands, listed in Table 5–2, to manage WebCenter connections. Configuration changes made using these WebCenter WLST commands are only effective after restarting the Managed Server on which the WebCenter application is deployed. For details, see Oracle Fusion Middleware Administrator’s Guide for Oracle WebCenter.
5.2.1 deleteConnection

Module: Oracle WebCenter

Use with WLST: Online

5.2.1.1 Description

Deletes a named WebCenter connection.

If you use deleteConnection to delete a WSRP or PDK-Java producer connection (instead of using deregisterWSRPProducer or deregisterPDKJavaProducer), unused secondary connections will remain, which you might want to remove. For example, when you delete a WSRP producer connection, its associated Web Service connection remains; when you delete a PDK-Java producer connection, its associated URL connection remains.

5.2.1.2 Syntax

deleteConnection(appName, name, [server, applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>name</td>
<td>Connection name.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.2.1.3 Example

The following example deletes a WebCenter connection.

```
  wls:/weblogic/serverConfig> deleteConnection(appName='webcenter',
       name='MyConnection')
```

5.3 Content Repository

Use the commands listed in Table 5–3 to manage content repository connections and configure the Documents service for a WebCenter application.

Configuration changes made using these WebCenter WLST commands are only effective after your restart the Managed Server on which the WebCenter application is deployed. For details, see Oracle Fusion Middleware Administrator’s Guide for Oracle WebCenter.
### 5.3.1 createJCRContentServerConnection

Module: Oracle WebCenter

Use with WLST: Online

#### 5.3.1.1 Description

Creates an Oracle Content Server repository connection, for a named WebCenter application.

#### 5.3.1.2 Syntax

```
createJCRContentServerConnection(appName, name, socketType, [url, serverHost, serverPort, keystoreLocation, keystorePassword, privateKeyAlias, privateKeyPassword, cacheInvalidationInterval, binaryCacheMaxEntrySize, adminUsername, adminPassword, extAppId, timeout, isPrimary, server, applicationVersion])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>appName</code></td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td><code>name</code></td>
<td>Connection name. The name must be unique (across all connection types) within the WebCenter application.</td>
</tr>
</tbody>
</table>
socketType | Specifies whether the Oracle Content Server connects on the content server listener port or the Web server filter, and whether the listener port is SSL enabled. Valid values are socket, web, and socketssl. This option has no default.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>socketType</td>
<td>Specifies whether the Oracle Content Server connects on the content server listener port or the Web server filter, and whether the listener port is SSL enabled. Valid values are socket, web, and socketssl. This option has no default.</td>
</tr>
<tr>
<td>url</td>
<td>Optional. Oracle Content Server URL. Required only if socketType is set to web. URL should be in the format: http://&lt;hostname&gt;:&lt;port&gt;/&lt;web root&gt;/&lt;plugin root&gt; For example, <a href="http://mycontentserver/cms/idcplg">http://mycontentserver/cms/idcplg</a>.</td>
</tr>
<tr>
<td>serverHost</td>
<td>Optional. Host name of the machine where the Oracle Content Server is running. Required if socketType is set to socket or socketssl.</td>
</tr>
</tbody>
</table>
| serverPort       | Optional. Port on which the Oracle Content Server listens. Required if socketType is set to socket or socketssl:  
  - Socket - Port specified for the incoming provider in the server.  
  - Socket SSL - Port specified for the sslincoming provider in the server.  
  For example, 4444 |
| keystoreLocation | Optional. Location of key store that contains the private key used to sign the security assertions. Required only if socketType is set to socketssl. The key store location must be an absolute path. |
| keystorePassword | Optional. Password required to access the key store. Required only if socketType is set to socketssl. |
| privateKeyAlias  | Optional. Client private key alias in the key store. The key is used to sign messages to the server. The public key corresponding to this private key must be imported in the server keystore. Required only if socketType is set to socketssl. The value for this argument must be a string that contains neither special characters nor white space. |
| privateKeyPassword | Optional. Password to be used with the private key alias in the key store. Required only if socketType is set to socketssl. |
| cacheInvalidationInterval | Optional. Polling interval (in minutes) used by the Oracle Content Server service provider interface (SPI) to check for cache invalidations. Defaults to 0 which means that cache invalidation is disabled. The minimum interval is 2 minutes. |
The following example creates a socket-based connection to an Oracle Content Server running on myhost.com at port 4444. For authentication purposes, an existing external application named myExtApp is used. See also, createExtAppConnection.

```wls:/weblogic/serverConfig> createJCRContentServerConnection(appName='webcenter', name='myContentServerConnection', socketType='socket', serverHost='myhost.com', serverPort='4444', extAppId='myExtApp', isPrimary='true')```

The following example creates an SSL socket-based connection to an Oracle Content Server repository.

```wls:/weblogic/serverConfig> createJCRContentServerConnection(appName='webcenter', name='myContentServerConnection', socketType='socketssl', serverHost='myhost.com', serverPort='4444', keystoreLocation='d:/keys/here', keystorePassword='AlphaSquad7', privateKeyAlias='enigma', privateKeyPassword='S0larPl3x1s', extAppId='myExtApp')```
5.3.2 setJCRContentServerConnection

Module: Oracle WebCenter
Use with WLST: Online

5.3.2.1 Description
Edits an existing Oracle Content Server connection. This command requires that you specify values for appName and name, plus one additional argument.

5.3.2.2 Syntax
setJCRContentServerConnection(appName, name, [socketType, url, serverHost, serverPort, keystoreLocation, privateKeyAlias, privateKeyPassword, cacheInvalidationInterval, binaryCacheMaxEntrySize, adminUsername, adminPassword, extAppId, timeout, isPrimary, server, applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>name</td>
<td>Name of an existing Oracle Content Server connection.</td>
</tr>
</tbody>
</table>
| socketType    | Optional. Specifies whether the Oracle Content Server connects on the content server listener port or the Web server filter, and whether the listener port is SSL enabled. Valid values are socket, web, and socketssl. This option has no default. Choose from:  
  - socket - Use an intradoc socket connection to connect to the Oracle Content Server. The client IP address must be added to the list of authorized addresses in the Oracle Content Server. In this case, the client is the machine on which Oracle WebCenter is running.  
  - socketssl - Use an intradoc socket connection to connect to the Oracle Content Server that is secured using the SSL protocol. The client's certificates must be imported in the server's trust store for the connection to be allowed. Because this is the most secure option, this is the recommended option whenever identity propagation is required (for example, in WebCenter Spaces).  
  - web - Use an HTTP(S) connection to connect to the Oracle Content Server. Note that for WebCenter Spaces, this option is not suitable for the back-end Oracle Content Server repository that is being used to store group space and personal space documents, because it does not allow identity propagation. |
| url           | Optional. Oracle Content Server URL. Required only if socketType is set to web. URL should be in the format: http://<hostname>:<port>/<web root>/<plugin root>  
  For example, http://mycontentserver/cms/idcplg. |
| serverHost    | Optional. Host name of the machine where the Oracle Content Server is running. Required if socketType is set to socket or socketssl. |
| serverPort    | Optional. Port on which the Oracle Content Server listens. Required if socketType is set to socket or socketssl:  
  - Socket - Port specified for the incoming provider in the server.  
  - Socket SSL - Port specified for the sslincoming provider in the server.  
  For example, 4444 |
5.3.2.3 Examples

The following example edits a socket-based connection to an Oracle Content Server.

```
wlw:/weblogic/serverConfig> setJCRContentServerConnection(appName='webcenter',
name='myContentServerConnection', socketType='socket',
```
The following example edits an SSL socket-based connection to an Oracle Content Server.

```java
wls:/weblogic/serverConfig>
serverHost='myhost.com', serverPort='4444',
extAppId='myExtApp', isPrimary='true')
```

The following example edits an SSL socket-based connection to an Oracle Content Server.

```java
wls:/weblogic/serverConfig>
serverHost='myhost.com', serverPort='4444',
extAppId='myExtApp', isPrimary='true')
```

### 5.3.3 listJCRContentServerConnections

**Module:** Oracle WebCenter  
**Use with WLST:** Online

#### 5.3.3.1 Description
Without any arguments, this command lists all of the Oracle Content Server connections that are configured for a named WebCenter application.

#### 5.3.3.2 Syntax

```
listJCRContentServerConnections(appName, [verbose], [name, server, applicationVersion])
```

#### Argument Definition

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>verbose</td>
<td>Optional. Displays content repository connection details in verbose mode. Valid options are true and false. When set to true, listJCRContentServerConnections lists all Oracle Content Server connections that are configured for a WebCenter application, along with their details. When set to false, only connection names are listed. This argument defaults to false.</td>
</tr>
<tr>
<td>name</td>
<td>Optional. Name of an existing Oracle Content Server connection. When specified you can view connection details for a specific Oracle Content Server connection. If you supply a value for name, you must supply a value for verbose.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

#### 5.3.3.3 Examples

The following example lists Oracle Content Server connections configured for an application named webcenter.

```
wls:/weblogic/serverConfig> listJCRContentServerConnections(appName='webcenter')
```

The following example lists all properties of the Oracle Content Server connection named myContentServerConnection1. The connection named
myContentServerConnection1 must exist and be an Oracle Content Server connection. If, for example, you specify an Oracle Portal connection, the properties are not listed and an error is displayed.

```
wls:/weblogic/serverConfig> listJCRContentServerConnections(appName='webcenter', verbose=true, name='myContentServerConnection1')
```

### 5.3.4 createJCRPortalConnection

**Module:** Oracle WebCenter

**Use with WLST:** Online

#### 5.3.4.1 Description

Creates an Oracle Portal repository connection.

#### 5.3.4.2 Syntax

```
createJCRPortalConnection(appName, name, dataSource, [extAppId, isPrimary, timeout, server, applicationVersion])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>name</td>
<td>Connection name. The name must be unique (across all connection types) within the WebCenter application.</td>
</tr>
<tr>
<td>dataSource</td>
<td>JNDI DataSource location used to connect to the portal. For example: jdbc/MyPortalDS. The datasource must be on the server where the WebCenter application is deployed.</td>
</tr>
<tr>
<td>extAppId</td>
<td>Optional. External application used to authenticate WebCenter users against Oracle Portal. This value should match the name of an existing external application connection. See also listExtAppConnections. If extAppId is not set, no change is made to the authentication method or external application ID. If extAppId is set to an empty string, the authentication method used is IDENTITY_PROPAGATION. With this method, the WebCenter application and Oracle Portal use the same identity store to authenticate users.</td>
</tr>
<tr>
<td>timeout</td>
<td>Optional. Length of time allowed to log in to Oracle Portal (in ms) before issuing a connection timeout message. If no timeout is set, there is no time limit for the login operation.</td>
</tr>
<tr>
<td>isPrimary</td>
<td>Optional. Valid string values are true and false. true specifies that this connection is the primary connection used by the Documents service. This argument defaults to false. In WebCenter Spaces, the primary connection must be an Oracle Content Server connection.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>
### 5.3.4.3 Example

The following example creates a Oracle Portal connection named `myPortalConnection` using the data source `jdbc/portalDS` and specifies that an external application, named `myExtApp`, is used for authentication.

```
wlsc:/weblogic/serverConfig> createJCRPortalConnection(appName='myApp', name='myPortalConnection', dataSource='jdbc/portalDS', extAppId='myExtApp', isPrimary='true')
```

### 5.3.5 setJCRPortalConnection

Module: Oracle WebCenter

Use with WLST: Online

#### 5.3.5.1 Description

Edits an existing Oracle Portal connection. This command requires that you specify values for either the `dataSource` or `isPrimary` argument.

#### 5.3.5.2 Syntax

```
setJCRPortalConnection(appName, name, [dataSource, extAppId, timeout, isPrimary, server, applicationVersion])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>name</td>
<td>Name of an existing Oracle Portal connection.</td>
</tr>
<tr>
<td>dataSource</td>
<td>Optional. JNDI DataSource location used to connect to the portal. For example: <code>jdbc/MyPortalDS</code>. The datasource must be on the server where the WebCenter application is deployed.</td>
</tr>
<tr>
<td>extAppId</td>
<td>Optional. External application used to authenticate WebCenter users against Oracle Portal. This value should match the name of an existing external application connection. See also <code>listExtAppConnections</code>. If <code>extAppId</code> is not set, no change is made to the authentication method or external application ID. If <code>extAppId</code> is set to an empty string, the authentication method used is <code>IDENTITY_PROPAGATION</code>. With this method, the WebCenter application and Oracle Portal use the same identity store to authenticate users.</td>
</tr>
<tr>
<td>timeout</td>
<td>Optional. Length of time allowed to log in to Oracle Portal (in ms) before issuing a connection timeout message. If no timeout is set, there is no time limit for the login operation.</td>
</tr>
<tr>
<td>isPrimary</td>
<td>Optional. Valid string values are <code>true</code> and <code>false</code>. <code>true</code> specifies that this connection is the primary connection used by the Documents service. When set to <code>false</code>, and the specified connection is the primary connection used by the Documents service, the primary connection is reset. If this parameter is not set, the primary connection used by the Documents service does not change. This argument has no default. In WebCenter Spaces, the primary connection must be an Oracle Content Server connection.</td>
</tr>
<tr>
<td>server</td>
<td></td>
</tr>
<tr>
<td>applicationVersion</td>
<td></td>
</tr>
</tbody>
</table>
### 5.3.5 Example

The following example edits Oracle Portal repository connection details.

```
wls:/weblogic/serverConfig> setJCRPortalConnection(appName='webcenter', name='myPortalConnection', dataSource='/newPortalDS', extAppId='myExtApp', isPrimary='false')
```

### 5.3.6 listJCRPortalConnections

**Module:** Oracle WebCenter  
**Use with WLST:** Online

#### 5.3.6.1 Description

Without any arguments, this command lists all of the Oracle Portal connections that are configured for a named WebCenter application.

#### 5.3.6.2 Syntax

```
listJCRPortalConnections(appName, [verbose, name, server, applicationVersion])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>appName</code></td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td><code>verbose</code></td>
<td>Optional. Displays content repository connection details in verbose mode. Valid options are <code>true</code> and <code>false</code>. When set to <code>true</code>, <code>listJCRPortalConnections</code> lists all Oracle Portal connections that are configured for a WebCenter application, along with their details. When set to <code>false</code>, only connection names are listed. This argument defaults to <code>false</code>.</td>
</tr>
<tr>
<td><code>name</code></td>
<td>Optional. Name of an existing Oracle Portal connection. When specified you can view connection details for a specific Oracle Portal connection. If you supply a value for <code>name</code>, you must supply a value for <code>verbose</code>.</td>
</tr>
<tr>
<td><code>server</code></td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, <code>WLS_Spaces</code>. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td><code>applicationVersion</code></td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

#### 5.3.6.3 Example

The following example lists all of the Oracle Portal connections that are configured for a WebCenter application.

```
wls:/weblogic/serverConfig> listJCRPortalConnections(appName='webcenter',
```

---

5-12 Oracle Fusion Middleware WebLogic Scripting Tool Command Reference
verbose=true, name='myPortalConnection')

5.3.7 createJCRFileSystemConnection
Module: Oracle WebCenter
Use with WLST: Online

5.3.7.1 Description
Creates a connection to a file system repository.

---

Note: File system connections must not be used in production or enterprise application deployments. This feature is provided for development purposes only.

---

5.3.7.2 Syntax
createJCRFileSystemConnection(appName, name, path, [isPrimary, server, applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>name</td>
<td>Connection name. The name must be unique (across all connection types) within the WebCenter application.</td>
</tr>
<tr>
<td>path</td>
<td>Full path to a folder whose contents you want to expose through this file system connection. For example, if you have a folder called C:\ProjectDocuments and you want to use that folder with the Documents service, you need to specify this folder as the path argument to this command.</td>
</tr>
<tr>
<td>isPrimary</td>
<td>Optional. Valid values are true and false. true specifies that this connection is the primary connection used by the Documents service. When set to false, and when the specified connection is the primary connection used by the Documents service, the primary connection is reset. If this parameter is not set, the primary connection used by the Documents service does not change. This argument has no default. In WebCenter Spaces, the primary connection must be an Oracle Content Server connection.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.3.7.3 Example
The following example creates a connection to a file system repository.

wls:/weblogic/serverConfig> createJCRFileSystemConnection(appName='webcenter', name='FSAConnection', path='C:/ProjectDocuments')

5.3.8 setJCRFileSystemConnection
Module: Oracle WebCenter
Use with WLST: Online

5.3.8.1 Description
Edits an existing file system repository connection. This command requires that you specify values for either the path or isPrimary arguments.

Note: File system connections must not be used in production or enterprise application deployments. This feature is provided for development purposes only.

5.3.8.2 Syntax
setJCRFileSystemConnection(appName, name, [path, [isPrimary, server, applicationVersion]])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Application name in which you want to set Document service properties.</td>
</tr>
<tr>
<td>name</td>
<td>Name for the connection to be used by the Documents service.</td>
</tr>
<tr>
<td>path</td>
<td>Optional. Full path to a folder whose contents you want to expose through this file system connection. For example, if you have a folder called C:\ProjectDocuments and you want to use that folder with the Documents service, you need to specify this folder as the path argument to this command.</td>
</tr>
<tr>
<td>isPrimary</td>
<td>Optional. Valid values are true and false. When set to true, specifies that this connection is the primary connection used by the Documents service. When set to false, and when the specified connection is the primary connection used by the Documents service, the primary connection is reset. If this parameter is not set, the primary connection used by the Documents service does not change. This argument has no default. Note that in WebCenter Spaces, the primary connection must be an Oracle Content Server connection.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.3.8.3 Example
The following example edits connection details for a file system repository.

wls:/weblogic/serverConfig> setJCRFileSystemConnection(appName='webcenter', name='FSAConnection', path='C:/ProjectDocuments')

5.3.9 listJCRFileSystemConnections
Module: Oracle WebCenter
Use with WLST: Online
5.3.9.1 Description
Without any arguments, this command lists all of the file system connections that are configured for a named WebCenter application.

**Note:** File system connections must not be used in production or enterprise application deployments. This feature is provided for development purposes only.

5.3.9.2 Syntax
```java
listJCRFileSystemConnections(appName, [verbose], [name, server, applicationVersion])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>verbose</td>
<td>Optional. Displays content repository connection details in verbose mode. Valid options are true and false. When set to true, <code>listJCRFileSystemConnections</code> lists all file system connections that are configured for a WebCenter application, along with their details. When set to false, only connection names are listed. This argument defaults to false.</td>
</tr>
<tr>
<td>name</td>
<td>Optional. Name of an existing file system connection. When specified you can view connection details for a specific file system connection. If you supply a value for name, you must supply a value for verbose.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.3.9.3 Examples
The following example lists all of the file system connections that are configured for an application named webcenter.
```
wlst:/weblogic/serverConfig> listJCRFileSystemConnections(appName='webcenter')
```

The following example lists all of the file system connections that are configured, in verbose mode.
```
wlst:/weblogic/serverConfig> listJCRFileSystemConnections(appName='webcenter', verbose=true)
```

5.3.10 `listDocumentsSpacesProperties`
Module: Oracle WebCenter
Use with WLST: Online
5.3.10.1 Description
Lists properties for the back-end Oracle Content Server repository that is being used by WebCenter Spaces to store group space and personal space documents. This command is only valid for the WebCenter Spaces application.

5.3.10.2 Syntax
listDocumentsSpacesProperties(appName, [server, applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter Spaces application in which you want to perform this operation—always webcenter.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.3.10.3 Example
The following example lists properties for the back-end Oracle Content Server repository that is being used by a WebCenter Spaces application (named webcenter) to store group space and personal space documents.

```
wls:/weblogic/serverConfig> listDocumentsSpacesProperties(appName='webcenter')
```

5.3.11 setDocumentsSpacesProperties
Module: Oracle WebCenter
Use with WLST: Online

5.3.11.1 Description
Modifies properties for the back-end Oracle Content Server repository that is being used by WebCenter Spaces to store group space data. This command is only valid for the WebCenter Spaces application.

5.3.11.2 Syntax
setDocumentsSpacesProperties(appName, [spacesRoot, adminUserName, applicationName, server, applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter Spaces application in which you want to perform this operation—always webcenter.</td>
</tr>
<tr>
<td>spacesRoot</td>
<td>Optional. Root folder under which group space content is stored. The value for this argument must use the format: /foldername. For example, /WebCenter or /WebCenterSpaces. The spacesRoot cannot be /, the root itself, and it must be unique across applications. If the folder specified does not exist it will be created for you. Note that if you provide a value for this argument, you must also provide values for the adminUserName and applicationName arguments.</td>
</tr>
</tbody>
</table>
5.3.11 Examples
The following example modifies connection properties for the back-end Oracle Content Server repository that is being used by WebCenter Spaces to store group space and personal space documents.

```
wls:/weblogic/serverConfig> setDocumentsSpacesProperties(appName='webcenter',
spacesRoot='/AccountingSpaces', adminUserName='admin',
applicationName='WCAccounting')
```

The following example modifies the administrator's user name for the back-end Oracle Content Server repository that is being used by WebCenter Spaces to store group space and personal space documents.

```
wls:/weblogic/serverConfig> setDocumentsSpacesProperties(appName='webcenter',
adminUserName='sysadmin')
```

5.3.12 deleteDocumentsSpacesProperties
Module: Oracle WebCenter
Use with WLST: Online

5.3.12.1 Description
Deletes properties for the back-end Oracle Content Server repository used by WebCenter Spaces, that is the adminUserName, applicationName, and spacesRoot. This command is only valid for the WebCenter Spaces application.

5.3.12.2 Syntax

```
deleteDocumentsSpacesProperties(appName, [server, applicationVersion])
```
The following example deletes connection properties (adminUserName, applicationName, spacesRoot) of the back-end Oracle Content Server repository that is being used by WebCenter Spaces.

```
> wls:/weblogic/serverConfig> deleteDocumentsSpacesProperties(appName='webcenter')
```

### 5.4 Discussions and Announcements

Use the commands listed in Table 5–4 to manage discussions server connections for WebCenter applications.

Configuration changes made using these WebCenter WLST commands are only effective after your restart the Managed Server on which the WebCenter application is deployed. For details, see Oracle Fusion Middleware Administrator’s Guide for Oracle WebCenter.

**Table 5–4 Discussion and Announcement WLST Commands**

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>createDiscussionForumConnection</td>
<td>Create a new discussions server connection for a WebCenter application.</td>
<td>Online</td>
</tr>
<tr>
<td>setDiscussionForumConnection</td>
<td>Edit an existing discussions server connection.</td>
<td>Online</td>
</tr>
<tr>
<td>setDiscussionForumConnectionProperty</td>
<td>Set an additional discussions server connection property.</td>
<td>Online</td>
</tr>
<tr>
<td>deleteDiscussionForumConnectionProperty</td>
<td>Delete a discussions server connection property.</td>
<td>Online</td>
</tr>
<tr>
<td>listDiscussionForumConnections</td>
<td>List all of the discussions server connections that are configured for an application.</td>
<td>Online</td>
</tr>
<tr>
<td>listDefaultDiscussionForumConnection</td>
<td>List the default discussions server connection for an application.</td>
<td>Online</td>
</tr>
<tr>
<td>setDefaultDiscussionForumConnection</td>
<td>Specify the default connection for the Discussions and Announcements services.</td>
<td>Online</td>
</tr>
<tr>
<td>setDiscussionForumServiceProperty</td>
<td>Specify defaults for the Discussions service.</td>
<td>Online</td>
</tr>
<tr>
<td>removeDiscussionForumServiceProperty</td>
<td>Remove defaults for the Discussions service.</td>
<td>Online</td>
</tr>
<tr>
<td>listDiscussionForumServiceProperties</td>
<td>List Discussions service properties.</td>
<td>Online</td>
</tr>
</tbody>
</table>
5.4.1 createDiscussionForumConnection

Module: Oracle WebCenter

Use with WLST: Online

5.4.1.1 Description

Creates a new discussions server connection for a named WebCenter application.

The Discussions service and the Announcements service both require a discussions server connection. Both services use the same discussions server connection.

While you can register multiple discussions server connections for a WebCenter application, only one connection is used for discussion and announcement services—the default (or active) connection.

5.4.1.2 Syntax

createDiscussionForumConnection(appName, name, url, adminUser, [secured, timeout, default, server, applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>name</td>
<td>Connection name. The name must be unique (across all connection types) within the WebCenter application.</td>
</tr>
<tr>
<td>url</td>
<td>URL of the discussions server hosting discussion forums and announcements. For example: <a href="http://myhost:8888/owc_discussions">http://myhost:8888/owc_discussions</a>.</td>
</tr>
<tr>
<td>adminUser</td>
<td>Name of the discussions server administrator. This account is used by the Discussions and Announcements services to perform administrative operations on behalf of WebCenter users. This account is mostly used for managing group space discussions and announcements in WebCenter Spaces. It is not necessary for this user to be a super admin. However, the user must have administrative privileges on the current application root category for WebCenter Spaces, that is, the category (on the discussions server) under which all group space discussions and announcements are stored.</td>
</tr>
</tbody>
</table>

Table 5–4 (Cont.) Discussion and Announcement WLST Commands

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>setAnnouncementServi ceProperty</td>
<td>Specify defaults for the Announcements service.</td>
<td>Online</td>
</tr>
<tr>
<td>removeAnnouncementServi ceProperty</td>
<td>Remove defaults for the Announcements service.</td>
<td>Online</td>
</tr>
<tr>
<td>listAnnouncementServi ceProperties</td>
<td>List Announcements service properties.</td>
<td>Online</td>
</tr>
</tbody>
</table>
5.4.1.3 Example
The following example creates a secure discussions server connection for a WebCenter application.

```
$wls:/weblogic/serverConfig> createDiscussionForumConnection(appName='webcenter',
name='MyDiscussionServer', url='http://myhost.com:8888/owc_discussions',
adminUser='admin', secured=true, default=false)
```

5.4.2 setDiscussionForumConnection
Module: Oracle WebCenter
Use with WLST: Online

5.4.2.1 Description
Edits an existing discussions server connection. Use this command to update connection attributes.

The connection is created using the `createDiscussionForumConnection` command.

5.4.2.2 Syntax
```
setDiscussionForumConnection(appName, name, [url, adminUser, secured, timeout, 
default, server, applicationVersion])
```
**Argument** | **Definition**
--- | ---
`appName` | Name of the WebCenter application in which you want to perform this operation.

`name` | Name of an existing discussions server connection.

`url` | Optional. URL to the discussions server.

`adminUser` | Optional. Name of the discussions server administrator. This account is used by the Discussions service to perform administrative operations on behalf of WebCenter users.

   This account is mostly used for managing group space discussions and announcements in WebCenter Spaces. It is not necessary for this user to be a super admin. However, the user must have administrative privileges on the current root category for WebCenter Spaces, that is, the category (on the discussions server) under which all WebCenter Spaces discussion forums are stored.

`secured` | Optional. Indicates that a secure communication (WS-Security) is required between the WebCenter application and the discussions server.

   This argument must be set to `true` as a secured connection is always required. The default is `false`.

   When set to `true` (secured mode), all WebService calls from the WebCenter application are sent with a user name token and client certificate. You must use the `setDiscussionForumConnectionProperty` command to configure WS-Security properties such as the keystore location, keystore password, and the encryption key details.

`timeout` | Optional. Length of time (in seconds) the Discussion and Announcement services wait for a response from the discussions server before issuing a connection timeout message. This argument defaults to `-1`. When set to `-1`, the service default (10 seconds) applies.

`default` | Optional. Indicates that this connection is the default connection for the Discussions and Announcements services. Required only if more than one connection is defined.

   Valid options are `true` and `false`. When set to `true`, the Discussion and Announcement services use this connection. When set to `false`, the connection is not used. The default is `false`.

   To specify that the Discussion and Announcements service use this connection, change the value from `false` to `true`.

   To disable this connection, use the `removeDiscussionForumServiceProperty` command:

   ```
   removeDiscussionForumServiceProperty('appName='webcenter', property='selected.connection')
   ```

   Note: While you can register multiple discussions server connections for a WebCenter application, only one connection is used for discussion and announcement services—the default (or active) connection.

`server` | Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`.

   Required when applications with the same name are deployed to different servers and also when you have a cluster.

`applicationVersion` | Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.
5.4.2.3 Example
The following example updates attributes for a secure discussions server connection named MyDiscussionsServer.

```
wlst:/weblogic/serverConfig> setDiscussionForumConnection(appName='webcenter',
name='MyDiscussionServer', url='http://myhost.com:7786/owc_discussions',
adminUser='admin', secured=true, default=true)
```

5.4.3 setDiscussionForumConnectionProperty
Module: Oracle WebCenter
Use with WLST: Online

5.4.3.1 Description
Sets a discussions server connection property. Use this command when additional parameters are required to connect to your discussions server, for example, to set WS-Security keystore and encryption properties for the connection.

This command provides an extensible way to add any connection property using a key and a value. (You are not limited to connection properties specified by createDiscussionForumConnection and setDiscussionForumConnection.)

**Note:** Do not use the setDiscussionForumConnectionProperty to set connection properties available through createDiscussionForumConnection or setDiscussionForumConnection. Attempting to do so, has no effect.

All known, additional connection properties are listed in Table 5–5, ”Additional Discussion Connection Properties”.

### Table 5–5 Additional Discussion Connection Properties

<table>
<thead>
<tr>
<th>Additional Connection Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| keystore.location             | Certificate file path in your local directory. Specify Windows paths as `<drive>:\<dir>\<file_name>` or `<drive>:\<dir>\<file_name>`. For example, to specify the Windows location `c:\keys\mykeystore.jks`, enter either:
  - `c:\keys\mykeystore.jks` or
  - `c:/keys/mykeystore.jks` |
| keystore.password             | Keystore password. To encrypt this password, set `secure=true`. |
| keystore.type                 | Keystore type associated with the certificate. Valid values are: jks (Java Key Store) and pks. |
| encryption.key.alias          | Key alias to be used for encryption. |
| encryption.key.password       | Password for accessing the encryption key. To encrypt this password, set `secure=true`. |
Discussions and Announcements

Oracle WebCenter Custom WLST Commands

5.4.3.2 Syntax

```
setDiscussionForumConnectionProperty(appName, name, key, value, [secure, server, applicationVersion])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>name</td>
<td>Name of an existing discussions server connection.</td>
</tr>
<tr>
<td>key</td>
<td>Name of the connection property.</td>
</tr>
<tr>
<td>value</td>
<td>Value for the property. Allows any property to be modified on the connection with a key and value.</td>
</tr>
<tr>
<td>secure</td>
<td>Optional. Indicates whether the property value must be stored securely using encryption. Valid options are true and false. When true, the value is encrypted. The default option is false. Set to true if you are storing passwords.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.4.3.3 Example

The following example configures the location of the keystore certificate for a discussions server connection named MyDiscussionServer.

```
wls:/weblogic/serverConfig> setDiscussionForumConnectionProperty
(appName='webcenter', name='MyDiscussionServer', key='keystore.location', value='c:\keys\mykeystore.jks')
```

The following example adds a custom discussions server connection property called myProperty1 with a value propertyValue1.

```
wls:/weblogic/serverConfig> setDiscussionForumConnectionProperty
(appName='webcenter', name='MyDiscussionServer', key='myProperty1',
```

Table 5–5 (Cont.) Additional Discussion Connection Properties

<table>
<thead>
<tr>
<th>Additional Connection Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>group.mapping</td>
<td>(WebCenter Spaces only) Determines whether a subcategory or a single forum is created on the discussions server for new group spaces. When set to forum (the default), a single forum is created under the application’s root category per group space. When set to category, a subcategory is created under the application root category per group space. When a subcategory that supports multiple forums is more suitable, use setDiscussionForumConnectionProperty to set the group.mapping property to category. If a group space template does not define the mapping (the Blank template, for example) then the group.mapping property is used. If there is no value in the template or the connection, the default setting is used (forum).</td>
</tr>
</tbody>
</table>
value='propertyValue1')

The following example adds a secured discussions server connection property called securedProperty with the value secureValue.

```
wlstalk:/weblogic/serverConfig> setDiscussionForumConnectionProperty
(appName='webcenter', name='MyDiscussionServer', key='securedProperty',
value='secureValue', secure=true)
```

### 5.4.4 deleteDiscussionForumConnectionProperty

Module: Oracle WebCenter

Use with WLST: Online

#### 5.4.4.1 Description

Deletes a discussions server connection property. Take care when deleting connection properties because the connection may not work as expected if the configuration becomes invalid as a result.

This command can only delete additional connection properties added using the setDiscussionForumConnectionProperty command.

#### 5.4.4.2 Syntax

```
deleteDiscussionForumConnectionProperty(appName, name, key, [server,
applicationVersion])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>name</td>
<td>Name of an existing discussions server connection.</td>
</tr>
<tr>
<td>key</td>
<td>Name of the connection property you want to delete.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

#### 5.4.4.3 Example

The following example deletes a discussions server connection property named myProperty1.

```
wlstalk:/weblogic/serverConfig> deleteDiscussionForumConnectionProperty
(appName='webcenter', name='MyDiscussionServer', key='myProperty1')
```

### 5.4.5 listDiscussionForumConnections

Module: Oracle WebCenter

Use with WLST: Online
5.4.5.1 Description
Lists all of the discussions server connections that are configured for a named WebCenter application.

5.4.5.2 Syntax
listDiscussionForumConnections(appName, [verbose, name, server, applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>verbose</td>
<td>Optional. Valid options are true and false. When set to true, listDiscussionForumConnections lists all of the discussions server connections that are configured for a WebCenter application, along with their details. When set to false, only connection names are listed. This argument defaults to false.</td>
</tr>
<tr>
<td>name</td>
<td>Optional. Name of an existing discussions server connection. Use this argument to view connection details for a specific discussions server connection.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.4.5.3 Examples
The following example lists the names of all of the discussions server connections that are currently configured for an application named webcenter.

wls:/weblogic/serverConfig>listDiscussionForumConnections(appName='webcenter')

The following example lists connection names and details for all of the discussions server connections currently configured for an application named webcenter.

wls:/weblogic/serverConfig>listDiscussionForumConnections(appName='webcenter', verbose=true)

The following example lists connection details for a discussions server connection named myDiscussionsServer.

wls:/weblogic/serverConfig> listDiscussionForumConnections(appName='webcenter', name='myDiscussionsServer')

5.4.6 listDefaultDiscussionForumConnection
Module: Oracle WebCenter
Use with WLST: Online

5.4.6.1 Description
Names the discussions server connection that the Discussions service and the Announcements service are using, in a named WebCenter application. While you can register multiple discussions server connections for a WebCenter application, the Discussions/Announcements service only uses one connection—known as the default (or active) connection.
5.4.6.2 Syntax

listDefaultDiscussionForumConnection(appName, [verbose, server, applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>verbose</td>
<td>Optional. Valid options are true and false. When set to true, the name and details of the discussions server connections are listed. When set to false, only the connection name displays. This argument defaults to false.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.4.6.3 Examples

The following example names the discussions server connection that the Discussions/Announcements service are using, in an application named webcenter.

```
wls:/weblogic/serverConfig>
listDefaultDiscussionForumConnection(appName='webcenter')
```

The following example lists the name and details of the discussions server connection that the Discussions/Announcements service are using.

```
wls:/weblogic/serverConfig>
listDefaultDiscussionForumConnection(appName='webcenter', verbose=true)
```

5.4.7 setDefaultDiscussionForumConnection

Module: Oracle WebCenter

Use with WLST: Online

5.4.7.1 Description

Specifies the default discussions server connection for the Discussions service and the Announcements service, in a named WebCenter application.

While you can register multiple discussions server connections with a WebCenter application, the Discussions/Announcements services only uses one connection—this is known as the default (or active) connection.

5.4.7.2 Syntax

setDefaultDiscussionForumConnection(appName, name, [server, applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>name</td>
<td>Name of an existing discussions server connection.</td>
</tr>
</tbody>
</table>
5.4.7.3 Example
The following example makes a connection named myDiscussionServer the default (or active) connection for the Discussions and Announcement services.

```
wlis:/weblogic/serverConfig> setDefaultDiscussionForumConnection
(appName='webcenter', name='myDiscussionServer')
```

5.4.8 setDiscussionForumServiceProperty
Module: Oracle WebCenter
Use with WLST: Online

5.4.8.1 Description
Specifies default values for the Discussions service.

Configurable properties for the Discussions service are listed in Table 5–6, "Discussion Service Configuration Properties".

### Table 5–6 Discussion Service Configuration Properties

<table>
<thead>
<tr>
<th>Configuration Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>topics.fetch.size</td>
<td>Maximum number of topics fetched by the Discussions service and displayed in the topics view.</td>
</tr>
<tr>
<td>forums.fetch.size</td>
<td>Maximum number of forums fetched by the Discussions service and displayed in the forums view.</td>
</tr>
<tr>
<td>recentTopics.fetch.size</td>
<td>Maximum number of topics fetched by the Discussions service and displayed in the recent topics view.</td>
</tr>
<tr>
<td>watchedTopics.fetch.size</td>
<td>Maximum number of topics fetched by the Discussions service and displayed in the watched topics view.</td>
</tr>
<tr>
<td>watchedForums.fetch.size</td>
<td>Maximum number of forums fetched by the Discussions service and displayed in the watched forums view.</td>
</tr>
<tr>
<td>application.root.category .id</td>
<td>Application root category ID on the discussions server under which all discussion forums are stored. For example, if set to 3, all forums are stored inside category 3.</td>
</tr>
</tbody>
</table>

5.4.8.2 Syntax
```
setDiscussionForumServiceProperty(appName, property, value, [server, applicationVersion])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>property</td>
<td>Name of the configuration property.</td>
</tr>
</tbody>
</table>
5.4.8.3 Example
The following example changes the default number of topics displayed in topics view.

```
wls:/weblogic/serverConfig> setDiscussionForumServiceProperty
  (appName='webcenter', property='topics.fetch.size', value='30')
```

5.4.9 removeDiscussionForumServiceProperty
Module: Oracle WebCenter
Use with WLST: Online

5.4.9.1 Description
Removes the current value that is set for a Discussions service property. Use this command to remove any of the properties listed in Table 5–6, "Discussion Service Configuration Properties".

Take care when using this command as removing values for these properties might cause unexpected behavior.

**Note:** Use this command syntax to disable the connection currently used for discussion and announcement services:

```
removeDiscussionForumServiceProperty('appName='webcenter', property='selected.connection')
```

This command forces the default connection argument to false. See also, setDiscussionForumConnection.

5.4.9.2 Syntax
```
removeDiscussionForumServiceProperty(appName, property, [server,
  applicationVersion])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>Value for the property.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>
5.4.9.3 Example
The following example clears the current topics.fetch.size property for the Discussions service, in an application named webcenter.

\[
\text{wls:/weblogic/serverConfig> removeDiscussionForumServiceProperty(appName='webcenter', property='topics.fetch.size')}
\]

5.4.10 listDiscussionForumServiceProperties
Module: Oracle WebCenter
Use with WLST: Online

5.4.10.1 Description
Lists all configurable properties for the Discussions service.

5.4.10.2 Syntax
\[
\text{listDiscussionForumServiceProperties(appName, [server, applicationVersion])}
\]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.4.10.3 Example
The following example lists configuration properties for the Discussions service, in an application named webcenter.

\[
\text{wls:/weblogic/serverConfig> listDiscussionForumServiceProperties(appName='webcenter')}
\]

5.4.11 setAnnouncementServiceProperty
Module: Oracle WebCenter
Use with WLST: Online

5.4.11.1 Description
Specifies default values for the Announcements service.

Configurable properties for the Announcements service are listed in Table 5–7, "Announcements Service Configuration Properties".

<table>
<thead>
<tr>
<th>Configuration Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>miniview.page_size</td>
<td>Maximum number of announcements displayed in the Announcements mini view.</td>
</tr>
</tbody>
</table>
### Table 5–7 (Cont.) Announcements Service Configuration Properties

<table>
<thead>
<tr>
<th>Configuration Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>mainview.page_size</code></td>
<td>Maximum number of announcements displayed in the Announcements main view.</td>
</tr>
<tr>
<td><code>linksview.page_size</code></td>
<td>Maximum number of announcements displayed in the Announcements links view.</td>
</tr>
<tr>
<td><code>announcements.expiration.days</code></td>
<td>Number of days that announcements display and remain editable.</td>
</tr>
</tbody>
</table>

#### 5.4.11.2 Syntax

```java
setAnnouncementServiceProperty(appName, property, value, [server, applicationVersion])
```

#### Argument Definition

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>appName</code></td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td><code>property</code></td>
<td>Name of the configuration property.</td>
</tr>
<tr>
<td><code>value</code></td>
<td>Property value.</td>
</tr>
<tr>
<td><code>server</code></td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, <code>WLS_Spaces</code>. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td><code>applicationVersion</code></td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

#### 5.4.11.3 Example

The following example changes the default number of days that announcements display, in an application named `webcenter`.

```
wlsc/weblogic/serverConfig> setAnnouncementServiceProperty(appName='webcenter', property='announcements.expiration.days', value='21')
```

### 5.4.12 removeAnnouncementServiceProperty

Module: Oracle WebCenter

Use with WLST: Online

#### 5.4.12.1 Description

Removes the current value that is set for an Announcements service property. Use this command to remove any of the properties listed in Table 5–7, "Announcements Service Configuration Properties".

Take care when using this command as removing values for these properties might cause unexpected behavior.

#### 5.4.12.2 Syntax

```java
removeAnnouncementServiceProperty(appName, property, [server, applicationVersion])
```
5.4.12.3 Example
The following example clears the `announcements.expiration.days` property for the Announcements service, in an application named `webcenter`.

```bash
wls:/weblogic/serverConfig> removeAnnouncementServiceProperty
(appName='webcenter', property='announcements.expiration.days')
```

5.4.13 listAnnouncementServiceProperties
Module: Oracle WebCenter
Use with WLST: Online

5.4.13.1 Description
Lists all configurable properties for the Announcements service.

5.4.13.2 Syntax
`listAnnouncementServiceProperties(appName, [server, applicationVersion])`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>appName</code></td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td><code>server</code></td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, <code>WLS_Spaces</code>. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td><code>applicationVersion</code></td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.4.13.3 Example
The following example lists configuration properties for the Announcements service, in an application named `webcenter`.

```bash
wls:/weblogic/serverConfig> listAnnouncementServiceProperties(appName='webcenter')
```

5.5 External Applications
Use the commands listed in Table 5–8 to manage external application connections for WebCenter applications.
Configuration changes made using these WebCenter WLST commands are immediately available in the WebCenter application.

**Table 5–8 External Application WLST Commands**

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>createExtAppConnection</td>
<td>Create an external application connection, for a named WebCenter application.</td>
<td>Online</td>
</tr>
<tr>
<td>setExtAppConnection</td>
<td>Edit an existing external application connection.</td>
<td>Online</td>
</tr>
<tr>
<td>listExtAppConnections</td>
<td>List individual or all external applications that are configured for a specific WebCenter application.</td>
<td>Online</td>
</tr>
<tr>
<td>addExtAppField</td>
<td>Add another login field for a specific external application connection.</td>
<td>Online</td>
</tr>
<tr>
<td>setExtAppField</td>
<td>Edit the value and display-to-user setting for a specific external application login field.</td>
<td>Online</td>
</tr>
<tr>
<td>removeExtAppField</td>
<td>Remove an application login field.</td>
<td>Online</td>
</tr>
<tr>
<td>addExtAppCredential</td>
<td>Specify shared or public credentials for an external application.</td>
<td>Online</td>
</tr>
<tr>
<td>setExtAppCredential</td>
<td>Edit shared or public credentials for an external application.</td>
<td>Online</td>
</tr>
<tr>
<td>removeExtAppCredential</td>
<td>Remove shared or public credentials currently configured for an external application.</td>
<td>Online</td>
</tr>
</tbody>
</table>

**5.5.1 createExtAppConnection**

Module: Oracle WebCenter

Use with WLST: Online

**5.5.1.1 Description**

Creates an external application connection, for a named WebCenter application.

**5.5.1.2 Syntax**

```text
createExtAppConnection(appName, name, [displayName, url, authMethod, userFieldName, pwdFieldName, server, applicationVersion])
```

**Argument Definition**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>name</td>
<td>Connection name. The name must be unique (across all connection types) within the WebCenter application.</td>
</tr>
<tr>
<td>displayName</td>
<td>Optional. External application display name. A user friendly name for the application that WebCenter users will recognize. The display name must be unique across all external applications within the WebCenter application.</td>
</tr>
<tr>
<td>url</td>
<td>Optional. External application login URL. To determine an application’s URL, navigate to the application’s login page and note down the URL for that page. For example: <a href="http://login.yahoo.com/config/login">http://login.yahoo.com/config/login</a></td>
</tr>
</tbody>
</table>
External Applications

Oracle WebCenter Custom WLST Commands

5.5.1.3 Example

The following example creates a connection for an external application named My Yahoo!, in a WebCenter application.

```
wlst:/weblogic/serverConfig> createExtAppConnection(appName='webcenter',
name='yahoo', displayName='My Yahoo!', url='http://login.yahoo.com/config/login',
authMethod='POST', userFieldName='login', pwdFieldName='passwd')
```

5.5.2 setExtAppConnection

Module: Oracle WebCenter
Use with WLST: Online

5.5.2.1 Description

Edits an existing external application connection.

5.5.2.2 Syntax

```
setExtAppConnection(appName, name, [displayName], [url], [authMethod],
[userFieldName], [pwdFieldName], [server, applicationVersion])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>name</td>
<td>Name of an existing external application connection.</td>
</tr>
<tr>
<td>displayName</td>
<td>Optional. External application display name. A user-friendly name for the application that WebCenter users will recognize.</td>
</tr>
</tbody>
</table>
5.5.2.3 Example
The following example updates the display name attribute for an external application named yahoo.

```
wlst:/weblogic/serverConfig> setExtAppConnection(appName='webcenter',
       name='yahoo', displayName='My Favorite Yahoo!')
```

5.5.3 listExtAppConnections
Module: Oracle WebCenter
Use with WLST: Online

5.5.3.1 Description
When used with only the appName argument, this command lists the names of all the external applications currently configured for a specific WebCenter application.

5.5.3.2 Syntax

```
listExtAppConnections(appName, [verbose, name, server, applicationVersion])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application for which you want to perform this operation.</td>
</tr>
</tbody>
</table>
### 5.5.3.3 Examples

The following example lists the names of all the external applications currently used by a WebCenter application named `webcenter`.

```
wlst:/weblogic/serverConfig> listExtAppConnections(appName='webcenter')
app1
app2
app3
```

The following example lists details for the external applications `app1`, `app2`, and `app3`.

```
wlst:/weblogic/serverConfig> listExtAppConnections(appName='webcenter', verbose=true)
----
app1
----
Name: app1
Display Name: Application1
Login URL: http://app1
Authentication Method: POST
User Field Name: login
Password Field Name: passwd
Shared Credential: Disabled
Public Credential: Disabled
----
app2
----
Name: app2
Display Name: Application2
Login URL: http://app2
Authentication Method: POST
User Field Name: login
Password Field Name: passwd
Additional Fields: {Account1:true, Account2:DefVal:false}
Shared Credential: Disabled
Public Credential: Enabled
----
app3
----
```

<table>
<thead>
<tr>
<th><strong>Argument</strong></th>
<th><strong>Definition</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>verbose</code></td>
<td>Optional. Displays external application details in verbose mode. Valid options are <code>true</code> and <code>false</code>. When set to <code>true</code>, <code>listExtAppConnections</code> lists all of the external applications that are configured for a WebCenter application, along with their details. When set to <code>false</code>, <code>listExtAppConnections</code> lists only the names of the external applications. This argument defaults to <code>false</code>. If you set this argument to <code>false</code>, do not specify the <code>name</code> argument.</td>
</tr>
<tr>
<td><code>name</code></td>
<td>Optional. Name of an existing external application connection. You can use this argument to view details about a specific connection.</td>
</tr>
<tr>
<td><code>server</code></td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, <code>WLS_Spaces</code>. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td><code>applicationVersion</code></td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>
Name: app3
Display Name: Application3
Authentication Method: POST
Shared Credential: Enabled
Public Credential: Enabled

The following example lists details for external application app1 only.

```
wlst:/weblogic/serverConfig> listExtAppConnections(appName='webcenter', verbose=true, name='app1')
----
app1
----
Name: app1
Display Name: Application1
Login URL: http://app1
Authentication Method: POST
User Field Name: login
Password Field Name: passwd
Shared Credential: Disabled
Public Credential: Disabled
```

5.5.4 addExtAppField

Module: Oracle WebCenter

Use with WLST: Online

5.5.4.1 Description

Adds another login field for a specific external application connection. For example, in addition to user name and password, an external application may require other login criteria such as Host and MailAddress.

Optionally, additional login fields can appear on the external application's login for a user to specify.

If you add another login field and the external application uses shared or public credentials, you can use the WLST commands and `setExtAppCredential` to update the shared/public credentials. See Section 5.5.7, "addExtAppCredential" and Section 5.5.8, "setExtAppCredential".

5.5.4.2 Syntax

```
addExtAppField(appName, name, fieldName, [fieldValue], [displayToUser], [server, applicationVersion])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>name</td>
<td>Name of an existing external application connection.</td>
</tr>
<tr>
<td>fieldName</td>
<td>Login field name. The name that identifies the field on the HTML login form. This field is not applicable if the application uses BASIC authentication.</td>
</tr>
<tr>
<td>fieldValue</td>
<td>Optional. Login field value. Enter a default value for the login field or leave blank for a user to specify. This argument is blank by default.</td>
</tr>
</tbody>
</table>
5.5.4.3 Example
This example creates an additional field named Account with the default value username.default.example in an external application called ABC. This field will be displayed in ABC’s login screen.

```bash
wls:/weblogic/serverConfig> addExtAppField(appName='webcenter', name='ABC', fieldName='Account', fieldValue='username.default.example', displayToUser=true)
```

5.5.5 setExtAppField
Module: Oracle WebCenter
Use with WLST: Online

5.5.5.1 Description
Modifies the field value and display-to-user setting for one or more login fields currently configured for an external application. Either fieldValue or displayToUser must be specified along with the external application name and login field name. The fieldValue and displayToUser arguments are optional.

Using this command has implications on any shared or public credentials that you might have created for this external application. If you modify displayToUser to true, you may also need to update existing shared user or public user credentials. See also Section 5.5.8, "setExtAppCredential".

5.5.5.2 Syntax
```bash
setExtAppField(appName, name, fieldName, [fieldValue], [displayToUser], [server, applicationVersion])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>name</td>
<td>Name of an existing external application connection.</td>
</tr>
<tr>
<td>fieldName</td>
<td>Name of an existing login field.</td>
</tr>
<tr>
<td>fieldValue</td>
<td>Optional. New or changed login field value. Enter a default value for the login field or leave blank for a user to specify. This argument is blank by default.</td>
</tr>
</tbody>
</table>
5.5.5.3 Example
The following example specifies a default value for a login field named Account and displays the field on the external application's credential provisioning screen.

```
wls://weblogic/serverConfig> setExtAppField(appName='webcenter', name='ABC',
fieldName='Account', fieldValue='admin', displayToUser=true)
```

5.5.6 removeExtAppField
Module: Oracle WebCenter
Use with WLST: Online

5.5.6.1 Description
Removes a login field from an external application connection.

This command has implications on any shared or public credentials that you may have created for this external application, that is, you may need to remove the login field from shared user or public user credentials.

You can use the `setExtAppCredential` command to remove a login field, if required. For example, external application `myApp` has an additional field called Account and public credentials were previously specified using:

```
addExtAppCredential(appName='webcenter', name='myApp', type='PUBLIC',
username='admin', password='mypublic.password', field='Account:admin@myhost.com')
```

If you remove the Account field, you can modify the credentials by running:

```
setExtAppCredential(appName='webcenter', name='myApp', type='PUBLIC',
username='admin', password='mypublic.password')
```

For details on using `setExtAppCredential`, see Section 5.5.8, "setExtAppCredential"

5.5.6.2 Syntax
```
removeExtAppField(appName, name, fieldName, [server, applicationVersion])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>appName</code></td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td><code>name</code></td>
<td>Connection name.</td>
</tr>
<tr>
<td><code>fieldName</code></td>
<td>Login field that you want to remove.</td>
</tr>
</tbody>
</table>
5.5.6.3 Example
The following example removes the additional login field named Account from an external application named ABC.

```sh
wls:/weblogic/serverConfig> removeExtAppField(appName='webcenter, name='ABC', fieldName='Account')
```

5.5.7 addExtAppCredential
Module: Oracle WebCenter
Use with WLST: Online

5.5.7.1 Description
Configures shared user or public user credentials for a specific external application.

When shared credentials are specified, every user accessing the WebCenter application is authenticated using the user name and password defined here. WebCenter users are not presented with a login form.

Public users accessing this application through WebCenter are logged in using the user name and password defined here.

If credentials already exists, a warning indicates that the setExtAppCredential command should be used instead.

5.5.7.2 Syntax
```
addExtAppCredential(appName, name, type, username, password, [field, server, applicationVersion])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>name</td>
<td>Name of an existing external application connection.</td>
</tr>
<tr>
<td>type</td>
<td>Credential type. Valid values are SHARED and PUBLIC.</td>
</tr>
<tr>
<td>username</td>
<td>Name of the shared or public user.</td>
</tr>
<tr>
<td>password</td>
<td>Password for the shared or public user.</td>
</tr>
<tr>
<td>field</td>
<td>Optional. Additional login field value. Use the format FieldName:FieldValue, where FieldName names an additional login field configured with displayToUser=true.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
</tbody>
</table>
5.5.7.3 Example
The following example specifies public credentials for an external application named ABC. The public user name is mypublic.username, the password is mypublic.password, and there is one additional field named Account:

```
wlst:/weblogic/serverConfig> addExtAppCredential(appName='webcenter', name='ABC', type='PUBLIC', username='mypublic.username', password='mypublic.password', field='Account:username.example')
```

5.5.8 setExtAppCredential
Module: Oracle WebCenter
Use with WLST: Online

**5.5.8.1 Description**
Modifies shared user or public user credentials currently configured for an external application. If the credential has already not been specified, then a warning indicates that `addExtAppCredential` needs to be used instead. See Section 5.5.7, "addExtAppCredential".

The arguments `username` and `password` are optional because `setExtAppCredential` only manipulates existing credentials.

You can use `setExtAppCredential` command to update passwords in systems that require changing passwords every few days.

**5.5.8.2 Syntax**

```
setExtAppCredential(appName, name, type, [username], [password], [field], [server, applicationVersion])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>name</td>
<td>Name of an existing external application connection.</td>
</tr>
<tr>
<td>type</td>
<td>Credential type. Valid values are SHARED and PUBLIC.</td>
</tr>
<tr>
<td>username</td>
<td>Optional. User name of the shared or public user.</td>
</tr>
<tr>
<td>password</td>
<td>Optional. Password for the shared or public user.</td>
</tr>
<tr>
<td>field</td>
<td>Optional. Additional login field value. Use the format FieldName:FieldValue, where FieldName names an additional login field configured with displayToUser=true.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>
5.5.8.3 Example
The following example changes the public user's login credentials for an external application named ABC.

```bash
wls:/weblogic/serverConfig> setExtAppCredential(appName='webcenter', name='ABC', type='PUBLIC', username='username.example', password='password.example', field='Account:username.example')
```

5.5.9 removeExtAppCredential
Module: Oracle WebCenter
Use with WLST: Online

5.5.9.1 Description
Removes shared user or public user credentials currently configured for an external application.
If credentials do not exist, an error displays.

5.5.9.2 Syntax
```bash
removeExtAppCredential(appName, name, type, [server, applicationVersion])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>name</td>
<td>Name of an existing external application connection.</td>
</tr>
<tr>
<td>type</td>
<td>Credential type. Valid values are SHARED and PUBLIC.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.5.9.3 Example
The following example removes shared credentials specified for an external application named ABC.

```bash
wls:/weblogic/serverConfig> removeExtAppCredential(appName='webcenter', name='ABC', type='SHARED')
```

5.6 Instant Messaging and Presence
Use the commands listed in Table 5-9, to manage instant messaging and presence server connections.
Configuration changes made using these WebCenter WLST commands are only effective after your restart the Managed Server on which the WebCenter application is deployed. For details, see Oracle Fusion Middleware Administrator’s Guide for Oracle WebCenter.
5.6.1 createIMPConnection

Module: Oracle WebCenter

Use with WLST: Online

5.6.1.1 Description

Creates an instant messaging and presence server connection for a named WebCenter application.

Use the listIMPAdapters command to find out which types of instant messaging and presence servers are supported. Out-of-the-box, WebCenter applications support Oracle WebLogic Communication Server (OWLCS) and Microsoft Live Communication Server (LCS).

While you can register multiple presence server connections for a WebCenter application, only one connection is used for instant messaging and presence services—the default (or active) connection.

5.6.1.2 Syntax

createIMPConnection(appName, name, adapter, url, domain, [appId, poolName, policyURI, timeout, default, server, applicationVersion])
## Argument Definition

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>name</td>
<td>Connection name. The name must be unique (across all connection types) within the WebCenter application.</td>
</tr>
<tr>
<td>url</td>
<td>URL of the server hosting instant messaging and presence services. For example: <a href="http://myowlcshost.com:8888">http://myowlcshost.com:8888</a></td>
</tr>
<tr>
<td>adapter</td>
<td>Adapter name. Specify the adapter that matches your instant messaging and presence server. Valid values are LCS and OWLCS. Choose LCS for Microsoft Live Communication Server. Choose OWLCS for Oracle WebLogic Communication Server.</td>
</tr>
<tr>
<td>domain</td>
<td>User domain associated with this connection. The domain specified is used to construct each user's SIP ID. For example, if the domain is oracle.com and presence is requested for user with name john then the SIP address resolved will be sip:<a href="mailto:joh@oracle.com">joh@oracle.com</a>. If the user SIP address needs to be resolved from the OID/LDAP server, then specify the user profile attribute that will provide the SIP address here as profile:&lt;attribute&gt; where profile is a keyword and attribute is the user profile attribute name where the SIP address is stored. For example, profile:primarySipAddress. SIP is short for Session Initiation Protocol - an Internet protocol for live communication between people.</td>
</tr>
<tr>
<td>appId</td>
<td>Optional. External application associated with the presence server connection. If specified, external application credential information is used to authenticate users against the LCS or OWLCS server. This argument is mandatory for LCS server connections. The external application you configure for the IMP service must use authMethod=POST, and specify an additional field with fieldName='Account' and displaytoUser=true. See also addExtAppField and setExtAppField.</td>
</tr>
<tr>
<td>poolName</td>
<td>Optional. (LCS connections only) Pool name that is required to create an LCS connection. Refer to Microsoft Live Communication Server documentation for details on the pool name. This argument is mandatory for LCS server connections.</td>
</tr>
<tr>
<td>policyURI</td>
<td>Optional. (OWLCS connections only) URI to the security policy that is required for authentication on the Oracle WebLogic Communication Server (OWLCS) server.</td>
</tr>
<tr>
<td>timeout</td>
<td>Optional. Length of time (in seconds) that the Instant Messaging and Presence service waits for a response from the presence server before issuing a connection timeout message. This argument defaults to -1. When set to -1, the service default (10 seconds) applies.</td>
</tr>
<tr>
<td>default</td>
<td>Optional. Indicates whether this connection is the default connection for the Instant Messaging and Presence service. Valid values are true and false. The default for this argument is false.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>
5.6.1.3 Examples
The following example creates an instant messaging and presence server connection to a Oracle WebLogic Communication Server named myOWLCSPresenceServer.

```wls:/weblogic/serverConfig>createIMPConnection(appName='webcenter',
name='myOWLCSPresenceServer', adapter='OWLCS',
url='http://myowlcshost.com:8888', domain='oracle.com')```

The following example also creates an instant messaging and presence server connection to a Oracle WebLogic Communication Server named myOWLCSPresenceServer.

```wls:/weblogic/serverConfig>createIMPConnection(appName='webcenter',
name='myOWLCSPresenceServer', adapter='OWCLS', url='http://myowlcshost.com:8888',
domain='oracle.com', policyURI='oracle/wss11_saml_token_with_message_protection_client_policy', timeout=60, default=false)```

The following example creates an instant messaging and presence server connection to a Microsoft Live Communication Server named myLCSPresenceServer.

```wls:/weblogic/serverConfig>createIMPConnection(appName='webcenter',
name='myLCSPresenceServer', adapter='LCS', url='http://mylcshost.com/owc/lcs',
domain='oracle.com', appId='LCSExtApp', poolName='pool1.myhost.com', timeout=60,
default=true)```

5.6.2 setIMPConnection
Module: Oracle WebCenter
Use with WLST: Online

5.6.2.1 Description
Edits an existing instant messaging and presence server connection. Use this command to update connection attributes.

The connection is created using the createIMPConnection command.

5.6.2.2 Syntax
setIMPConnection(appName, name, [adapter, url, domain, appId, poolName,
policyURI, timeout, default, server, applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>name</td>
<td>Name of an existing presence server connection.</td>
</tr>
<tr>
<td>adapter</td>
<td>Optional. Adapter name. Specify the adapter that matches your instant messaging and presence server. Valid values are LCS and OWLCS. Choose LCS for Microsoft Live Communication Server. Choose OWLCS for Oracle WebLogic Communication Server.</td>
</tr>
<tr>
<td>url</td>
<td>Optional. URL of the server hosting instant messaging and presence services.</td>
</tr>
</tbody>
</table>
**Instant Messaging and Presence**

The domain specified is used to construct each user's SIP ID. For example, if the domain is `oracle.com` and presence is requested for user with name `john` then the SIP address resolved will be `sip:john@oracle.com`.

If the user SIP address needs to be resolved from the OID/LDAP server, then specify the user profile attribute that will provide the SIP address here as `profile:<attribute>` where `profile` is a keyword and `<attribute>` is the user profile attribute name where the SIP address is stored. For example, `profile:primarySipAddress`.

Alternatively, dynamically resolve the user SIP address on your own using a custom class. Provide a custom class that implements `oracle.webcenter.collab.rtc.IMPAddressResolver` and then specify the custom class here as `custom:com.company.custom.AddressResolver` (assuming your class is `com.company.custom.AddressResolver`).

SIP is short for Session Initiation Protocol - an Internet protocol for live communication between people.

---

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>domain</td>
<td>Optional. User domain associated with this connection. The domain specified is used to construct each user's SIP ID. For example, if the domain is <code>oracle.com</code> and presence is requested for user with name <code>john</code> then the SIP address resolved will be <code>sip:john@oracle.com</code>. If the user SIP address needs to be resolved from the OID/LDAP server, then specify the user profile attribute that will provide the SIP address here as <code>profile:&lt;attribute&gt;</code> where <code>profile</code> is a keyword and <code>&lt;attribute&gt;</code> is the user profile attribute name where the SIP address is stored. For example, <code>profile:primarySipAddress</code>. Alternatively, dynamically resolve the user SIP address on your own using a custom class. Provide a custom class that implements <code>oracle.webcenter.collab.rtc.IMPAddressResolver</code> and then specify the custom class here as <code>custom:com.company.custom.AddressResolver</code> (assuming your class is <code>com.company.custom.AddressResolver</code>). SIP is short for Session Initiation Protocol - an Internet protocol for live communication between people.</td>
</tr>
<tr>
<td>appId</td>
<td>Optional. External application associated with the presence server connection. If specified, external application credential information is used to authenticate users against the LCS or OWLCS server. This argument is mandatory for LCS server connections. The external application you configure for the IMP service must use <code>authMethod=POST</code>, and specify an additional field with <code>fieldName='Account'</code> and <code>displayToUser=true</code>. See also <code>addExtAppField</code> and <code>setExtAppField</code>.</td>
</tr>
<tr>
<td>poolName</td>
<td>Optional. (LCS connections only) Pool name that is required to create an LCS connection. Mandatory for LCS connections. Refer to Microsoft Live Communication Server documentation for details on the pool name.</td>
</tr>
<tr>
<td>policyURI</td>
<td>Optional. (OWLCS connections only) URI to the security policy that is required for authentication on the Oracle WebLogic Communication Server (OWLCS) server.</td>
</tr>
<tr>
<td>timeout</td>
<td>Optional. Length of time (in seconds) that the Instant Messaging and Presence service waits for a response from the presence server before issuing a connection timeout message. This argument defaults to -1. When set to -1, the service default (10 seconds) applies.</td>
</tr>
<tr>
<td>default</td>
<td>Optional. Indicates whether this connection is the default connection for the Instant Messaging and Presence service. Valid values are <code>true</code> and <code>false</code>. The default for this argument is <code>false</code>. To specify that the Instant Messaging and Presence service uses this connection, change the value from <code>false</code> to <code>true</code>. To disable this connection, use the <code>removeIMPServiceProperty</code> command: <code>removeIMPServiceProperty('appName='webcenter', property='selected.connection')</code> While you can register multiple presence server connections for a WebCenter application, only one connection is used for instant messaging and presence services—the default (or active) connection.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, <code>WLS_Spaces</code>. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
</tbody>
</table>
5.6.2.3 Examples

The following example updates attributes for an existing instant messaging and presence server connection.

```bash
wls:/weblogic/serverConfig>
setIMPConnection(appName='webcenter',
    name='myOWLCSPresenceServer', adapter='OWCLS', url='http://myowlcshost.com:8888',
    domain='oracle.com')
```

The following example sets attributes on an existing instant messaging and presence server connection.

```bash
wls:/weblogic/serverConfig>
setIMPConnection(appName='webcenter',
    name='myOWLCSPresenceServer', adapter='OWCLS', url='http://myowlcshost.com:8888',
    domain='oracle.com', policyURI='oracle/wss11_saml_token_with_message_protection_client_policy', timeout=60, default=false)
```

The following example sets attributes on an existing instant messaging and presence server connection.

```bash
wls:/weblogic/serverConfig>
setIMPConnection(appName='webcenter',
    name='myLCSPresenceServer', adapter='LCS', url='http://mylcshost.com/owc/lcs',
    domain='oracle.com', appId='LCSExtApp', poolName='pool1.myhost.com', timeout=60,
    default=false)
```

5.6.3 setIMPConnectionProperty

Module: Oracle WebCenter

Use with WLST: Online

5.6.3.1 Description

Sets an instant messaging and presence server connection property. Use this command if additional parameters are required to connect to your presence server. This is an extensible way to add any connection property using a key and a value. (You are not limited to connection properties specified by createIMPConnection and setIMPConnection.)

All known, additional connection properties are listed in Table 5–10, "Additional IMP Connection Properties".

<table>
<thead>
<tr>
<th>Additional Connection Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>presence.url</code> (OWLCS only) URL to the OWLCS Presence service. Required if the OWLCS Presence service is deployed on a separate node. When no value is specified, the connection <code>url</code> property is used.</td>
<td></td>
</tr>
<tr>
<td><code>contacts.url</code> (OWLCS only) URL to the OWLCS Contact Management service. Required if the OWLCS Contact Management service is deployed on a separate node. When no value is specified, the connection <code>url</code> property is used.</td>
<td></td>
</tr>
</tbody>
</table>
Do not use the `setIMPConnectionProperty` to set connection properties available through `createIMPConnection` or `setIMPConnection`. Attempting to do so, has no effect.

### 5.6.3.2 Syntax

```
setIMPConnectionProperty(appName, name, key, value, [secure, server, applicationVersion])
```

### Table 5–10 (Cont.) Additional IMP Connection Properties

<table>
<thead>
<tr>
<th>Additional Connection Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>call.url</code></td>
<td>(OWLCS only) URL to the OWLCS Third Party Call service. Required if the OWLCS Third Party Call service is deployed on a separate node. When no value is specified, the connection’s <code>url</code> property is used.</td>
</tr>
<tr>
<td><code>call.method</code></td>
<td>(OWLCS only) Third party call method. Valid values are: <code>sip</code> and <code>pstn</code>. When set to <code>sip</code>, third party calls are forwarded to Oracle Communicators. When set to <code>pstn</code>, calls are forwarded to PSTN telephones (<code>contact.number.attribute</code> provides the phone number).</td>
</tr>
<tr>
<td><code>call.domain</code></td>
<td>(OWLCS only) Domain name of the PSTN gateway. Required when the <code>call.method</code> is <code>pstn</code>. If no domain name is supplied, the connection’s <code>domain</code> value is used.</td>
</tr>
<tr>
<td><code>contact.number.attribute</code></td>
<td>(OWLCS only) User profile attribute used to store users’ phone numbers. The default attribute is <code>BUSINESS_PHONE</code>. Required when the <code>call.method</code> is <code>pstn</code>.</td>
</tr>
<tr>
<td><code>primary.domain</code></td>
<td>(OWLCS and LCS) User domain. If WebCenter user names are qualified with a domain, specify that domain here. For example, when user names are <code>xyz@example.com</code>, the <code>primary.domain</code> is <code>example.com</code>.</td>
</tr>
</tbody>
</table>
5.6.3 Example
The following example adds a custom instant messaging and presence server connection property called admin.user with a default value admin.

```
WLS:/weblogic/serverConfig> setIMPConnectionProperty(appName='webcenter',
name='MyLCSPresenceServer', key='admin.user', value='admin')
```

5.6.4 deleteIMPConnectionProperty
Module: Oracle WebCenter
Use with WLST: Online

5.6.4.1 Description
Deletes an instant messaging and presence server connection property. Use caution when deleting connection properties because the connection might not work as expected if the configuration becomes invalid as a result.

This command can only delete additional connection properties added using the setIMPConnectionProperty command.

5.6.4.2 Syntax
```
deleteIMPConnectionProperty(appName, name, key, [server, applicationVersion])
```

5.6.4.3 Example
The following example deletes an instant messaging and presence server connection property named admin.user.

```
WLS:/weblogic/serverConfig> deleteIMPConnectionProperty(appName='webcenter',
name='MyLCSPresenceServer', key='admin.user')
```

5.6.5 listIMPAdapters
Module: Oracle WebCenter
Use with WLST: Online

5.6.5.1 Description
Lists which types of instant messaging and presence servers Oracle WebCenter supports. Out-of-the-box, WebCenter applications support Oracle WebLogic Communication Server (OWLCS) and Microsoft Live Communication Server (LCS).
5.6.5.2 Syntax
listIMPAdapters()

5.6.5.3 Example
The following example lists which presence servers are supported.

    wls:/weblogic/serverConfig> listIMPAdapters()

5.6.6 listIMPConnections
Module: Oracle WebCenter
Use with WLST: Online

5.6.6.1 Description
Lists all of the instant messaging and presence server connections that are configured for a named WebCenter application.

5.6.6.2 Syntax
listIMPConnections(appName, [verbose], [name], [server, applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>verbose</td>
<td>Optional. Displays presence server connection details in verbose mode. Valid values are true and false. When set to true, listIMPConnections lists all of the presence server connections that are configured for a WebCenter application, along with their details. When set to false, only connection names are listed. This argument defaults to false.</td>
</tr>
<tr>
<td>name</td>
<td>Optional. Name of an existing presence server connection. Use this argument to view connection details for a specific presence server connection. Note that if you use the name argument when verbose argument set to true, the verbose argument is ignored.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.6.6.3 Examples
The following example lists all of the instant messaging and presence server connections that are configured for an application named webcenter.

    wls:/weblogic/serverConfig> listIMPConnections(appName='webcenter')

The following example lists all of the instant messaging and presence server connections that are configured for the application in verbose mode.

    wls:/weblogic/serverConfig> listIMPConnections(appName='webcenter', verbose=true)
The following example lists connection details for an instant messaging and presence server connections named impConnection1.

```
wlsw/weblogic/serverConfig> listIMPConnections(appName='webcenter', name='impConnection1')
```

### 5.6.7 listDefaultIMPConnection

**Module:** Oracle WebCenter

**Use with WLST:** Online

#### 5.6.7.1 Description

Lists the connection that the Instant Messaging and Presence service is using, in a named WebCenter application. While you can register multiple presence server connections for a WebCenter application, the Instant Messaging and Presence service only uses one connection —the default (or active) connection.

If only one presence server connection is available, that connection is assumed to be the default connection.

#### 5.6.7.2 Syntax

```
listDefaultIMPConnection(appName, verbose, [server, applicationVersion])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>verbose</td>
<td>Optional. Displays the default presence server connection in verbose mode, if available. Valid options are true and false. When set to true, the name and details of the presence server connection are listed. When set to false, only the connection name displays. This argument defaults to false.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

#### 5.6.7.3 Example

The following example lists the name and details of the connection that the Instant Messaging and Presence service is using, in an application named webcenter.

```
wls:/weblogic/serverConfig> listDefaultIMPConnection(appName='webcenter', verbose=true)
```

### 5.6.8 setDefaultIMPConnection

**Module:** Oracle WebCenter

**Use with WLST:** Online

#### 5.6.8.1 Description

Specifies the default connection for the Instant Messaging and Presence service, in a named WebCenter application. While you can register multiple presence server
connections with a WebCenter application, the Instant Messaging and Presence service only uses one connection — the default (or active) connection.

If only one presence server connection is available, that connection is assumed to be the default connection.

**5.6.8.2 Syntax**

```plaintext
setDefaultIMPConnection(appName, name, [server, applicationVersion])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>name</td>
<td>Name of an existing instant messaging and presence connection.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

**5.6.8.3 Example**
The following example makes a connection named `myPresenceServer` the default (or active) connection for the Instant Messaging and Presence service.

```plaintext
wls:/weblogic/serverConfig>setDefaultIMPConnection(appName='webcenter', name='myPresenceServer')
```

**5.6.9 setIMPServiceProperty**

Module: Oracle WebCenter

Use with WLST: Online

**5.6.9.1 Description**

Specifies default values for the Instant Messaging and Presence service.

Configurable properties for the Instant Messaging and Presence service are listed in Table 5–11, "Instant Messaging and Presence Service Configuration Properties".

<table>
<thead>
<tr>
<th>Configuration Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>selected.connection</td>
<td>Connection used by the Instant Messaging and Presence service.</td>
</tr>
<tr>
<td>rtc.cache.time</td>
<td>Cache timeout for instant messaging and presence data. The default is 60 seconds.</td>
</tr>
</tbody>
</table>
Table 5–11  (Cont.) Instant Messaging and Presence Service Configuration Properties

<table>
<thead>
<tr>
<th>Configuration Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>resolve.display.name.from.user.profile</td>
<td>Determines what to display if user display names are missing. When set to false, and display name information is unavailable, only the user name displays in the application. When set to true, and display name information is unavailable, display names are read from user profile data. Setting this option to true will impact performance. The default setting is false. Display names are not mandatory in presence data. If the WebCenter application does not always provide display names by default and you consider this information important, set resolve.display.name.from.user.profile to true so that display names always display.</td>
</tr>
</tbody>
</table>

5.6.9.2 Syntax

setIMPServiceProperty(appName, property, value, [server, applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>property</td>
<td>Name of the configuration property.</td>
</tr>
<tr>
<td>value</td>
<td>Value for the property.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.6.9.3 Example

The following example changes the default cache timeout for instant messaging and presence data, in an application named webcenter.

wls:/weblogic/serverConfig> setIMPServiceProperty(appName='webcenter', property='rtc.cache.time', value='30')

5.6.10 removeIMPServiceProperty

Module: Oracle WebCenter
Use with WLST: Online

5.6.10.1 Description

Removes the current value that is set for an Instant Messaging and Presence service property. Use this command to remove any of the properties listed in Table 5–11, "Instant Messaging and Presence Service Configuration Properties".

Take care when using this command as removing values for these properties might cause unexpected behavior.
**Note:** Use this command syntax to disable the connection currently used by the Instant Messaging and Presence service:

```java
removeIMPServiceProperty('appName='webcenter',
property='selected.connection')
```

This command forces the default connection argument to false. See also, `setIMPConnection`.

### 5.6.10.2 Syntax

```java
removeIMPServiceProperty(appName, property, [server, applicationVersion])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>property</td>
<td>Name of the configuration property.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

### 5.6.10.3 Example

The following example clears the default cache expiration value for the Instant Messaging and Presence service, in an application named `webcenter`.

```bash
wls:/weblogic/serverConfig> removeIMPServiceProperty(appName='webcenter',
property='rtc.cache.time')
```

### 5.6.11 listIMPServiceProperties

Module: Oracle WebCenter

Use with WLST: Online

#### 5.6.11.1 Description

Lists all configurable properties for the Instant Messaging and Presence service.

#### 5.6.11.2 Syntax

```java
listIMPServiceProperties(appName, [server, applicationVersion])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>
5.6.11.3 Example
The following example lists configuration properties for the Instant Messaging and Presence service, in an application named webcenter.

\[ \text{wls:/weblogic/serverConfig> listIMPServiceProperties(appName='webcenter')} \]

5.7 Mail

Use the commands listed in Table 5–12 to manage mail server connections for a WebCenter application.

You can register multiple mail server connections:

- **WebCenter Spaces** supports multiple mail connections. The mail connection configured with `default=true` is the default connection for mail services in WebCenter Spaces. All additional connections are offered as alternatives; WebCenter Spaces users can choose which one they want to use through user preferences.

- **Custom WebCenter applications** only use one mail connection—the connection configured with `default=true`. Any additional connections are ignored.

Configuration changes made using these WebCenter WLST commands are only effective after your restart the Managed Server on which the WebCenter application is deployed. For details, see Oracle Fusion Middleware Administrator’s Guide for Oracle WebCenter.

**Table 5–12 Mail WLST Commands**

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>createMailConnection</td>
<td>Create a mail server connection for a WebCenter application.</td>
<td>Online</td>
</tr>
<tr>
<td>setMailConnection</td>
<td>Edit an existing mail server connection.</td>
<td>Online</td>
</tr>
<tr>
<td>setMailConnectionProperty</td>
<td>Set mail server connection properties.</td>
<td>Online</td>
</tr>
<tr>
<td>deleteMailConnectionProperty</td>
<td>Delete a mail server connection property.</td>
<td>Online</td>
</tr>
<tr>
<td>listMailConnections</td>
<td>List all of the mail server connections that are configured for an application.</td>
<td>Online</td>
</tr>
<tr>
<td>listDefaultMailConnection</td>
<td>List the default mail server connection that is configured for an application.</td>
<td>Online</td>
</tr>
<tr>
<td>setDefaultMailConnection</td>
<td>Set a specified connection as the default mail server connection.</td>
<td>Online</td>
</tr>
<tr>
<td>setMailServiceProperty</td>
<td>Specify defaults for the Mail service.</td>
<td>Online</td>
</tr>
<tr>
<td>removeMailServiceProperty</td>
<td>Remove defaults for the Mail service.</td>
<td>Online</td>
</tr>
<tr>
<td>listMailServiceProperties</td>
<td>List Mail service properties.</td>
<td>Online</td>
</tr>
</tbody>
</table>

5.7.1 createMailConnection

Module: Oracle WebCenter

Use with WLST: Online
5.7.1.1 Description

Creates a mail server connection for a WebCenter application.

WebCenter applications support the Microsoft Exchange Server or any mail server that supports IMAP4 and SMTP. The most important mail server connection attributes are: imapHost, imapPort, imapSecured, smtpHost, smtpPort, and smtpSecured

You can register multiple mail server connections:

- **WebCenter Spaces** supports multiple mail connections. The mail connection configured with default=true is the default connection for mail services in WebCenter Spaces. All additional connections are offered as alternatives; WebCenter Spaces users can choose which one they want to use through user preferences.

- **Custom WebCenter applications** only use one mail connection—the connection configured with default=true. Any additional connections are ignored.

5.7.1.2 Syntax

createMailConnection(appName, name, imapHost, imapPort, smtpHost, smtpPort, imapSecured, smtpSecured, appId, [timeout, default, server, applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>name</td>
<td>Connection name. The name must be unique (across all connection types) within the WebCenter application.</td>
</tr>
<tr>
<td>imapHost</td>
<td>Host name of the machine on which the IMAP service is running.</td>
</tr>
<tr>
<td>imapPort</td>
<td>Port on which the IMAP service listens.</td>
</tr>
<tr>
<td>smtpHost</td>
<td>Host name of the machine where the SMTP service is running.</td>
</tr>
<tr>
<td>smtpPort</td>
<td>Port on which the SMTP service listens.</td>
</tr>
<tr>
<td>imapSecured</td>
<td>Optional. Specifies whether the mail server connection to the IMAP server is SSL-enabled. Valid values are true and false. The default for this argument is false.</td>
</tr>
<tr>
<td>smtpSecured</td>
<td>Optional. Specifies whether the SMTP server is secured. Valid values are true and false. The default for this argument is false.</td>
</tr>
<tr>
<td>appId</td>
<td>External application associated with the mail server connection.</td>
</tr>
<tr>
<td>timeout</td>
<td>Optional. Length of time (in seconds) that the service waits to acquire a connection before terminating. This argument defaults to -1. When set to -1, the service default (10 seconds) applies.</td>
</tr>
</tbody>
</table>

External application credential information is used to authenticate users against the IMAP and SMTP servers. The same credentials are supplied to authenticate the user on both the IMAP and SMTP servers. See also createExtAppConnection.

The external application you configure for the Mail service must use authMethod=POST, and specify an additional field with fieldName='Email Address' and displayToUser=true. See also addExtAppField and setExtAppField.
Mail

5.7.1.3 Examples

The following example creates a mail server connection named `myMailConnection` and enables the mail server connection in the WebCenter application.

```
  wls:/weblogic/serverConfig> createMailConnection(appName='webcenter',
         name='myMailConnection', imapHost='myimaphost.com', imapPort=143,
         smtpHost='mysmtpHost.com', smtpPort=25, imapSecured=false, smtpSecured=false,
         appId='extApp_Mail', timeout=60, default=true)
```

5.7.2 setMailConnection

Module: Oracle WebCenter

Use with WLST: Online

5.7.2.1 Description

Edits an existing mail connection. Use this command to update connection attributes.

The connection is created using the `createMailConnection` command.

(WebCenter Spaces application only.) This command enables you to set additional, optional, LDAP server attributes that cannot be set using `createMailConnection`. When LDAP details are defined, the Mail service creates, edits, and deletes group space distribution lists for WebCenter Spaces. Group space distribution lists are named after their group space (excluding non-java identifiers) and assigned a domain (derived from the `domain` attribute, for example, `@mycompany.com`). If LDAP details are not provided, group space distribution lists are not created or maintained. The mail server must be a `Microsoft Exchange Server`.

5.7.2.2 Syntax

```
  setMailConnection(appName, name, [imapHost, imapPort, smtpHost, smtpPort,
         imapSecured, smtpSecured, appId, default, ldapHost, ldapPort, ldapBaseDN,
         ldapAdminUser, ldapAdminPassword, ldapSecured, domain, defaultUser, timeout,
         server, applicationVersion])
```
<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>name</td>
<td>Name of an existing mail server connection.</td>
</tr>
<tr>
<td>imapHost</td>
<td>Optional. Host name of the machine on which the IMAP service is running.</td>
</tr>
<tr>
<td>imapPort</td>
<td>Optional. Port on which the IMAP service listens.</td>
</tr>
<tr>
<td>smtpHost</td>
<td>Optional. Host name of the machine where the SMTP service is running.</td>
</tr>
<tr>
<td>smtpPort</td>
<td>Optional. Port on which the SMTP service listens.</td>
</tr>
<tr>
<td>imapSecured</td>
<td>Optional. Specifies whether the connection to the IMAP server is secured (SSL-enabled). Valid values are true and false. The default for this argument is false.</td>
</tr>
<tr>
<td>smtpSecured</td>
<td>Optional. Specifies whether the connection to the SMTP server is secured (SSL-enabled). Valid values are true and false. The default for this argument is false.</td>
</tr>
<tr>
<td>appId</td>
<td>Optional. External application associated with the mail server connection. External application credential information is used to authenticate users against the IMAP and SMTP servers. The same credentials are supplied to authenticate the user on both the IMAP and SMTP servers. See also <code>createExtAppConnection</code>. The external application you configure for the Mail service must use authMethod=POST, and specify an additional field with fieldName='Email Address' and displaytoUser=true. See also <code>addExtAppField</code> and <code>setExtAppField</code>.</td>
</tr>
<tr>
<td>ldapHost</td>
<td>Optional. Host name of the machine where the LDAP directory server is running.</td>
</tr>
<tr>
<td>ldapPort</td>
<td>Optional. Port on which the LDAP directory server listens.</td>
</tr>
<tr>
<td>ldapBaseDN</td>
<td>Optional. Base distinguished name for the LDAP schema. For example, CN=Users, DC=oracle, DC=com.</td>
</tr>
<tr>
<td>ldapAdminUser</td>
<td>Optional. User name of the LDAP directory server administrator. A valid administrator with privileges to make entries into the LDAP schema.</td>
</tr>
<tr>
<td>ldapAdminPassword</td>
<td>Optional. Password for the LDAP directory server administrator. This password will be stored in a secured store.</td>
</tr>
<tr>
<td>ldapSecured</td>
<td>Optional. Specifies whether the connection to the LDAP server is secured (SSL enabled). Valid values are true and false. The default for this argument is false. Set this to true for all LDAP communications over SSL.</td>
</tr>
<tr>
<td>domain</td>
<td>Optional. Domain name appended to group space distribution lists. For example, if the domain attribute is set to mycompany.com, the Finance Project group space will maintain a distribution list named <a href="mailto:FinanceProject@oracle.com">FinanceProject@oracle.com</a>.</td>
</tr>
<tr>
<td>defaultUser</td>
<td>Optional. Comma-delimited list of user names to whom you want to grant moderation capabilities. These users become members of every group space distribution list that is created. The users specified must exist in the Base LDAP schema (specified in the <code>ldapBaseDN</code> argument).</td>
</tr>
</tbody>
</table>
### 5.7.2.3 Examples

The following example sets individual attributes of a mail server connection.

```plaintext
wls:/weblogic/serverConfig> setMailConnection(appName='webcenter',
name='myMailConnection', imapHost='myimaphost.com', imapPort=143,
smtpHost='mysmtpHost.com', smtpPort=25, imapSecured=false, smtpSecured=false,
appId='extApp_Mail', timeout=60, default=true)
```

The following example sets individual attributes of a mail server connection.

```plaintext
wls:/weblogic/serverConfig> setMailConnection(appName='webcenter',
name='myMailConnection', imapPort=993, imapSecured=true, smtpPort=465,
smtpSecured=true)
```

The following example sets LDAP attributes for a mail server connection.

```plaintext
wls:/weblogic/serverConfig> setMailConnection(appName='webcenter',
name='myMailConnection', domain='ORACLE.COM', defaultUser='admin',
imapHost='myimaphost.com', imapPort=143, smtpHost='mysmtpHost.com',
imapSecured=false, smtpSecured=false, smtpPort=25, appId='extApp_Mail',
default=true, ldapHost='myldaphost.com', ldapPort=389,
ldapBaseDN='CN=Users,DC=exchange,DC=uk,DC=com', ldapAdminUser='administrator',
ldapAdminPassword='adminpswd', ldapSecure=false, timeout=60)
```

### 5.7.3 setMailConnectionProperty

**Module:** Oracle WebCenter

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>timeout</td>
<td>Optional. Length of time (in seconds) that the service waits to acquire a connection before terminating. This argument defaults to -1. When set to -1, the service default (10 seconds) applies.</td>
</tr>
<tr>
<td>default</td>
<td>Optional. Indicates whether this connection is the default (or active) connection for the Mail service. Valid values are true and false. This argument defaults to false. true specifies that this connection is the default connection for the Mail service.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>
5.7.3.1 Description
Sets a mail server connection property. Use this command if additional parameters are required to connect to your mail server. This is an extensible way to add any connection property using a key and a value. (You are not limited to connection properties specified by `createMailConnection` and `setMailConnection`.)

All known, additional connection properties are listed in Table 5–13, "Additional Mail Connection Properties".

<table>
<thead>
<tr>
<th>Additional Connection Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>charset</td>
<td>Characterset used on the connection. The default charset is UTF-8. To use a different characterset, such as ISO-8859-1, set the charset connection property.</td>
</tr>
</tbody>
</table>

**Note:** Do not use the `setMailConnectionProperty` to set connection properties available through `createMailConnection` or `setMailConnection`. Attempting to do so, has no effect.

5.7.3.2 Syntax

```
setMailConnectionProperty(appName, name, key, value, [secure], [server], [applicationVersion])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>name</td>
<td>Name of an existing mail server connection.</td>
</tr>
<tr>
<td>key</td>
<td>Name of the connection property.</td>
</tr>
<tr>
<td>value</td>
<td>Value for the property. Allows any property to be modified on the connection with a key and value.</td>
</tr>
<tr>
<td>secure</td>
<td>Optional. Indicates whether the property value must be stored securely using encryption. Valid options are true and false. When true, the value is encrypted. The default option is false.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
</tbody>
</table>
5.7.3 Example
The following example adds a custom mail server connection property called myProperty1 with a default value propertyValue1.

```
wlstalk/weblogic/serverConfig> setMailConnectionProperty(appName='webcenter', name='myMailServer', key='myProperty1', value='propertyValue1')
```

5.7.4 deleteMailConnectionProperty
Module: Oracle WebCenter
Use with WLST: Online

5.7.4.1 Description
Deletes a mail server connection property. Take care when deleting connection properties because the connection may not work as expected if the configuration becomes invalid as a result.

This command can only delete additional connection properties added using the `setMailConnectionProperty` command.

5.7.4.2 Syntax
```
deleteMailConnectionProperty(appName, name, key, [server], [applicationVersion])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>appName</code></td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td><code>name</code></td>
<td>Name of an existing mail server connection.</td>
</tr>
<tr>
<td><code>key</code></td>
<td>Name of the connection property you want to delete.</td>
</tr>
<tr>
<td><code>server</code></td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td><code>applicationVersion</code></td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.7.4.3 Example
The following example deletes a mail server connection property named mailProperty1.

```
wls:/weblogic/serverConfig> deleteMailConnectionProperty(appName='webcenter', name='myMailServer', key='mailProperty1')
```

5.7.5 listMailConnections
Module: Oracle WebCenter
Use with WLST: Online
5.7.5.1 Description
Lists all of the mail server connections that are configured for a named WebCenter application.

5.7.5.2 Syntax
listMailConnection(appName, [verbose, name, server, applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>verbose</td>
<td>Optional. Displays mail server connection details in verbose mode. Valid options are true and false. When set to true, listMailConnections lists all of the mail server connections that are configured for a WebCenter application, along with their details. When set to false, only connection names are listed. This argument defaults to false.</td>
</tr>
<tr>
<td>name</td>
<td>Optional. Name of an existing mail server connection. Use this argument to view connection details for a specific mail server connection.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.7.5.3 Example
The following example lists the names of mail server connections that are currently configured for an application named webcenter.

```
wlsc:/weblogic/serverConfig> listMailConnections(appName='webcenter')
```

The following example lists connection names and details for all of the mail server connections that are currently configured for an application named webcenter.

```
wlsc:/weblogic/serverConfig> listMailConnections(appName='webcenter', verbose=true)
```

The following example lists connection details for a mail server connection named mailConnection1.

```
wls:/weblogic/serverConfig> listMailConnections(appName='webcenter', name='mailConnection1')
```

5.7.6 listDefaultMailConnection
Module: Oracle WebCenter
Use with WLST: Online

5.7.6.1 Description
Lists the default mail server connection that the Mail service is using, in a named WebCenter application.
You can register multiple mail server connections but there can only be one default connection:

- **WebCenter Spaces** supports multiple mail connections. The mail connection configured with `default=true` is the default connection for mail services in WebCenter Spaces. All additional connections are offered as alternatives; WebCenter Spaces users can choose which one they want to use through user preferences.

- **Custom WebCenter applications** only use one mail connection—the connection configured with `default=true`. Any additional connections are ignored.

### 5.7.6.2 Syntax

`listDefaultMailConnection(appName, [verbose], [server], [applicationVersion])`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>appName</code></td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td><code>verbose</code></td>
<td>Optional. Displays the default mail server connection in verbose mode, if available. Valid options are <code>true</code> and <code>false</code>. When set to <code>true</code>, the name and details of the mail server connection are listed. When set to <code>false</code>, only the connection name displays. This argument defaults to <code>false</code>.</td>
</tr>
<tr>
<td><code>server</code></td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, <code>WLS_Spaces</code>. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td><code>applicationVersion</code></td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

### 5.7.6.3 Example

The following example lists the name and details of the mail server connection that the Mail service is using, in an application named `webcenter`.

```
  wls:/weblogic/serverConfig> listDefaultMailConnection(appName='webcenter', verbose=true)
```

### 5.7.7 setDefaultMailConnection

Module: Oracle WebCenter

Use with WLST: Online

#### 5.7.7.1 Description

Specifies the `default` mail server connection for the Mail service, in a named WebCenter application.

You can register multiple mail server connections but there can only be one default connection:

- **WebCenter Spaces** supports multiple mail connections. The mail connection configured with `default=true` is the default connection for mail services in WebCenter Spaces. All additional connections are offered as alternatives; WebCenter Spaces users can choose which one they want to use through user preferences.
Custom WebCenter applications only use one mail connection—the connection configured with default=true. Any additional connections are ignored.

5.7.7.2 Syntax

setDefaultMailConnection(appName, name, [server], [applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>name</td>
<td>Name of an existing mail connection.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.7.7.3 Example

The following example configures the Mail service to use a connection named myMailServer.

wls:/weblogic/serverConfig>setDefaultMailConnection(appName='webcenter', name='myMailServer')

5.7.8 setMailServiceProperty

Module: Oracle WebCenter
Use with WLST: Online

5.7.8.1 Description

Specifies default values for the Mail service.

Configurable properties for the Mail service are listed in Table 5–14, "Mail Service Configuration Properties".

Table 5–14 Mail Service Configuration Properties

<table>
<thead>
<tr>
<th>Configuration Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mail.messages.fetch.size</td>
<td>Maximum number of messages displayed in mail inboxes.</td>
</tr>
<tr>
<td>resolve.email.address.to.name</td>
<td>Determines whether user email addresses are resolved to WebCenter user names when LDAP is configured. Valid values are true and false. The default value is false. When set to true, WebCenter user names display instead of email addresses in Mail task flows. Set this property to true if the Instant Messaging and Presence service requires user names to obtain presence status because presence information cannot be obtained when the Mail service provides email addresses. Setting this value to true does impact application performance so you must take this into consideration when setting this property.</td>
</tr>
</tbody>
</table>
5.7.8.2 Syntax

```
setMailServiceProperty(appName, property, value, [server], [applicationVersion])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>property</td>
<td>Name of the configuration property</td>
</tr>
<tr>
<td>value</td>
<td>Value for the property.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.7.8.3 Example

The following example increases the default number of messages displayed in mail inboxes to 100, in an application named webcenter.

```
wls:/weblogic/serverConfig> setMailServiceProperty('appName='webcenter',
property='mail.messages.fetch.size', value='100')
```

5.7.9 removeMailServiceProperty

Module: Oracle WebCenter
Use with WLST: Online

5.7.9.1 Description

Removes the current value that is set for a Mail service property. Use this command to remove any of the properties listed in Table 5–14, "Mail Service Configuration Properties".

Take care when using this command as removing values for these properties might cause unexpected behavior.

---

**Note:** Use this command syntax to stop the Mail service from using the current default connection:

```
removeMailServiceProperty('appName='webcenter',
property='selected.connection')
```

This command forces the default connection argument to false. See also, `setMailConnection`.

---

5.7.9.2 Syntax

```
removeMailServiceProperty(appName, property, [server], [applicationVersion])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>property</td>
<td>Name of the configuration property</td>
</tr>
</tbody>
</table>
5.7.9.3 Example
The following example clears the current mail.messages.fetch.size setting for the Mail service, in an application named webcenter.

```
> removeMailServiceProperty(appName='webcenter', property='mail.messages.fetch.size')
```

5.7.10 listMailServiceProperties
Module: Oracle WebCenter
Use with WLST: Online

**5.7.10.1 Description**
Lists all configurable properties for the Mail service.

**5.7.10.2 Syntax**
```
listMailServiceProperties(appName, [server], [applicationVersion])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

**5.7.10.3 Example**
The following example lists configuration properties for the Mail service, in an application named webcenter.

```
> listMailServiceProperties(appName='webcenter')
```

5.8 Personal Events

Use the commands listed in Table 5–15 to manage personal events server connections for a WebCenter application.

Configuration changes made using these WebCenter WLST commands are only effective after your restart the Managed Server on which the WebCenter application is deployed. For details, see Oracle Fusion Middleware Administrator’s Guide for Oracle WebCenter.
5.8.1 createPersonalEventConnection

Module: Oracle WebCenter

Use with WLST: Online

5.8.1.1 Description

Creates a personal events server connection for a named WebCenter application.


While you can register multiple personal events connections for a WebCenter application, only one connection is used for personal events services - the default (or active) connection.

5.8.1.2 Syntax

createPersonalEventConnection(appName, name, webServiceUrl, adapterName, appId, [default, server, applicationVersion])

Table 5–15 Personal Events WLST Commands

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>createPersonalEventConnection</td>
<td>Create a personal events server connection for a named WebCenter application.</td>
<td>Online</td>
</tr>
<tr>
<td>setPersonalEventConnection</td>
<td>Edit an existing personal events server connection.</td>
<td>Online</td>
</tr>
<tr>
<td>listPersonalEventConnections</td>
<td>List all of the personal events server connections that are configured for a named WebCenter application</td>
<td>Online</td>
</tr>
</tbody>
</table>

**Argument** | **Definition**
--- | ---
appName | Name of the WebCenter application in which you want to perform this operation. For WebCenter Spaces, the name is always webcenter.
name | Connection name. The name must be unique (across all connection types within the WebCenter application.
webServiceUrl | URL of the Web service exposing the event application.
Use the format <protocol>://<host>:<port>/<appWebServiceInterface>://<WSName>
adapterName | Specify the adapter that matches the personal events server. Valid values are MSEx2003 and MSEx2007. Choose MSEx2003 for Microsoft Exchange Server 2003 and MSEx2007 for Microsoft Exchange Server 2007.
Each adapter has its own mechanism of authenticating and exchanging data
appId | External application associated with the Microsoft Exchange Server providing personal events services. If specified, external application credential information is used to authenticate users against the Microsoft Exchange Server.
5.8.1.3 Example

The following example creates a connection named MyPEConnection for WebCenter Spaces (appName='webcenter'). The connection points to a Microsoft Exchange Server 2007 and is designated as the default connection for the Personal Events service.

```wls:/weblogic/serverConfig>createPersonalEventConnection(appName='webcenter', name='MyPEConnection', webServiceUrl='http://myexchange.com/EWS/Services.wsdl', adapterName='MSEx2007', appId='ExtPEApp', default=true)
```

The following example creates a connection named MyPEConnection for a WebCenter Spaces. The connection points to a Microsoft Exchange Server 2003.

```wls:/weblogic/serverConfig>createPersonalEventConnection(appName='webcenter', name='MyPEConnection', webServiceUrl='http://myexchange.com/ExchangeWS/PersonalEventsWebService.asmx', adapterName='MSEx2003', appId='ExtPEApp')
```

5.8.2 setPersonalEventConnection

Module: Oracle WebCenter
Use with WLST: Online

5.8.2.1 Description

Edits a personal events server connection for a named WebCenter application.

5.8.2.2 Syntax

```setPersonalEventConnection(appName, name, [webServiceUrl, adapterName, appId, default, server, applicationVersion])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation. For WebCenter Spaces, the name is always webcenter.</td>
</tr>
<tr>
<td>name</td>
<td>Name of an existing personal events server connection.</td>
</tr>
</tbody>
</table>

Argument Definition

default
Optional. Indicates whether this connection is the default connection for the Personal Events service. Valid values are true and false. The default for this argument is false.
To specify that the Personal Events service uses this connection, set the value to true.
While you can register multiple connections for a WebCenter application, only one connection is used for personal event services—the default (or active) connection.

server
Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces.
Required when applications with the same name are deployed to different servers and also when you have a cluster.

applicationVersion
Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.
5.8.2 Example
The following example updates the Web service URL for a connection named MyPEConnection.

```java
wls:/weblogic/serverConfig>setPersonalEventConnection(appName='webcenter',
name='MyPEConnection', webServiceUrl='http://myexchange.com/EWS/Services.wsdl')
```

The following example makes a connection named MyPEConnection the default connection for personal event services in WebCenter Spaces.

```java
wls:/weblogic/serverConfig>setPersonalEventConnection(appName='webcenter',
name='MyPEConnection', default=true)
```

5.8.3 listPersonalEventConnections
Module: Oracle WebCenter
Use with WLST: Online

5.8.3.1 Description
Lists all of the personal events server connections that are configured for a named WebCenter application.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>webServiceUrl</td>
<td>Optional. URL of the Web service exposing the event application. Use the format <code>&lt;protocol&gt;://&lt;host&gt;:&lt;port&gt;/&lt;appWebServiceInterface&gt;/&lt;WSName&gt;</code></td>
</tr>
<tr>
<td>adapterName</td>
<td>Optional. Specify the adapter that matches the personal events server. Valid values are MSEx2003 and MSEx2007. Choose MSEx2003 for Microsoft Exchange Server 2003 and MSEx2007 for Microsoft Exchange Server 2007. Each adapter has its own mechanism of authenticating and exchanging data</td>
</tr>
<tr>
<td>appId</td>
<td>Optional. External application associated with the Microsoft Exchange Server providing personal events services. If specified, external application credential information is used to authenticate users against the Microsoft Exchange Server.</td>
</tr>
<tr>
<td>default</td>
<td>Optional. Indicates whether this connection is the default connection for the Personal Events service. Valid values are true and false. The default for this argument is false. To specify that the Personal Events service uses this connection, set the value to true. While you can register multiple connections for a WebCenter application, only one connection is used for personal event services—the default (or active) connection.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>
5.8.3.2 Syntax

```
listPersonalEventConnections(appName, [verbose, name, server, applicationVersion])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>verbose</td>
<td>Optional. Displays connection details for the Personal Events service in verbose mode. Valid options are true and false. When set to true, listPersonalEventConnections lists all of the personal events server connections that are configured for a WebCenter application, along with their details. When set to false, only connection names are listed. This argument defaults to false. When set to false, do not specify the name argument.</td>
</tr>
<tr>
<td>name</td>
<td>Optional. Name of an existing personal events connection. Use this argument to view connection details for a specific personal events server.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.8.3.3 Example

The following example lists connection names and details for all of the personal events server connections currently configured for WebCenter Spaces.

```
wls:/weblogic/serverConfig>listPersonalEventConnections(appName='webcenter', verbose=true)
```

The following example displays connection details for a personal events server connection named MyPEConnection.

```
wls:/weblogic/serverConfig>listPersonalEventConnections(appName='webcenter', verbose=true, name='MyPEConnection')
```

5.9 Portlet Producers

Use the commands listed in Table 5–16 to manage portlet producers used in WebCenter applications.

All configuration changes made using these WebCenter WLST commands are immediately available in the WebCenter application.

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>registerWSRPProducer</td>
<td>Create and register a WSRP producer.</td>
<td>Online</td>
</tr>
<tr>
<td>setWSRPProducer</td>
<td>Edit WSRP producer registration details.</td>
<td>Online</td>
</tr>
<tr>
<td>listWSRPProducers</td>
<td>List WSRP producer registration details.</td>
<td>Online</td>
</tr>
<tr>
<td>deregisterWSRPProducer</td>
<td>Deregister a WSRP producer, and delete the associated WSRP and Web Service connections.</td>
<td>Online</td>
</tr>
</tbody>
</table>
Table 5–16 (Cont.) Producer WLST Commands

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>listWSRPProducerRegistrationProperties</td>
<td>List registration properties supported by a WSRP producer.</td>
<td>Online</td>
</tr>
<tr>
<td>listWSRPProducerUserCategories</td>
<td>List any user categories that the WSRP producer might support.</td>
<td>Online</td>
</tr>
<tr>
<td>mapWSRPProducerUserCategory</td>
<td>Map a role that is defined in the specified application to a user category supported by a WSRP producer.</td>
<td>Online</td>
</tr>
<tr>
<td>registerPDKJavaProducer</td>
<td>Create and register an Oracle PDK-Java producer.</td>
<td>Online</td>
</tr>
<tr>
<td>setPDKJavaProducer</td>
<td>Edit PDK-Java producer registration details.</td>
<td>Online</td>
</tr>
<tr>
<td>deregisterPDKJavaProducer</td>
<td>Deregister an Oracle PDK-Java producer, deleting the associated connection.</td>
<td>Online</td>
</tr>
<tr>
<td>listPDKJavaProducers</td>
<td>List registered Oracle PDK-Java producers.</td>
<td>Online</td>
</tr>
<tr>
<td>refreshProducer</td>
<td>Refresh the metadata stored for the named producer to reflect the portlets currently offered by that producer.</td>
<td>Online</td>
</tr>
<tr>
<td>registerOOTBProducers</td>
<td>Register out-of-the-box producers provided with Oracle WebCenter.</td>
<td>Online</td>
</tr>
<tr>
<td>deregisterOOTBProducers</td>
<td>Deregister out-of-the-box producers provided with Oracle WebCenter.</td>
<td>Online</td>
</tr>
<tr>
<td>registerSampleProducers</td>
<td>Register the sample producers provided with Oracle WebCenter.</td>
<td>Online</td>
</tr>
<tr>
<td>deregisterSampleProducers</td>
<td>Deregister sample producers.</td>
<td>Online</td>
</tr>
</tbody>
</table>

5.9.1 registerWSRPProducer

Module: Oracle WebCenter

Use with WLST: Online

5.9.1.1 Description

Creates a connection to a WSRP portlet producer and registers the WRSP producer with a named WebCenter application. When you create a WSRP producer connection, a Web Service connection is also created named <name>-wsconn where <name> is the value specified for the name argument.

5.9.1.2 Syntax

registerWSRPProducer(appName, name, url, [proxyHost], [proxyPort], [timeout],[externalApp],[registrationProperties],[tokenType],[issuer],[defUser],[keyStorePath],[keyStorePswd],[sigKeyAlias],[sigKeyPswd],[encKeyAlias],[encKeyPswd],[recptAlias],[server],[applicationVersion])
Portlet Producers

Oracle WebCenter Custom WLST Commands

### Argument Definition

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>name</strong></td>
<td>Connection name. The name must be unique (across all connection types) within the WebCenter application. The name you specify here will appear in the Oracle Composer (under the Portlets folder).</td>
</tr>
</tbody>
</table>
| **url**      | Producer WSDL URL. The syntax will vary according to your WSRP implementation, for example:  

   - `http://host_name:port_number/context_root/portlets/wsrp2?WSDL`
   - `http://host_name:port_number/context_root/portlets/wsrp1?WSDL`
   - `http://host_name:port_number/context_root/portlets/?WSDL` (WSRP 1.0 for backward compatibility)

   Where:
   - `host_name` is the server where your producer is deployed
   - `port_number` is the HTTP listener port number
   - `context_root` is the Web application’s context root
   - `portlets[/wsrp(1|2)]?WSDL` is static text. The text entered here depends on how the producer is deployed.

   For example:

| **proxyHost** | Optional. Host name or IP address of the proxy server. A proxy is required when the WebCenter application and the remote portlet producer are separated by a firewall and an HTTP proxy is needed to communicate with the producer. |
| **proxyPort** | Optional. Port number on which the proxy server listens. |
| **timeout**   | Optional. Timeout setting for communications with the producer, in seconds. For example, the maximum time the producer may take to register, deregister, or display portlets on WebCenter pages. This argument defaults to 30. Individual portlets may define their own timeout period, which takes precedence over the value expressed here. |
| **registrationProperties** | Optional. A list of registration properties and their values. The format of this argument must be a comma-separated list of valid registration properties, each followed by an equals symbol and the value. For example: `name=Producer, key=123`. The registration properties for a producer can be found using `listWSPProducerRegistrationProperties`. See Section 5.9.5, "listWSPProducerRegistrationProperties". |
Portlet Producers

Argument & Definition

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>tokenType</strong></td>
<td>Optional. Type of token profile to use for authentication with this WSRP producer. Valid values are:</td>
</tr>
<tr>
<td></td>
<td>- <strong>USERNAME_WITHOUT_PASSWORD</strong> (oracle/wss10_username_id_propagation_with_msg_protection_client_policy)—This policy provides message protection (integrity and confidentiality) and identity propagation for outbound SOAP requests in accordance with the WS-Security 1.0 standard. Credentials <em>(username only)</em> are included in outbound SOAP request messages through a WS-Security UsernameToken header. No password is included. Message protection is provided using WS-Security 1.0’s Basic128 suite of asymmetric key technologies. Specifically, RSA key mechanisms for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption.</td>
</tr>
<tr>
<td></td>
<td>- <strong>USERNAME_WITH_PASSWORD</strong> (oracle/wss10_username_token_with_message_protection_client_policy)—This policy provides message protection (integrity and confidentiality) and authentication for outbound SOAP requests in accordance with the WS-Security v1.0 standard. Both plain text and digest mechanisms are supported. This policy uses WS-Security’s Basic 128 suite of asymmetric key technologies. Specifically, RSA key mechanism for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption. Use this token profile if the WSRP producer has a different identity store. You will need to define an external application pertaining to the producer and associate the external application with this producer.</td>
</tr>
<tr>
<td></td>
<td>- <strong>SAML_TOKEN_WITH_MSG_INTEGRITY</strong> (wss10_saml_token_with_message_integrity_client_policy)—This policy provides message-level integrity protection and SAML-based authentication for outbound SOAP requests in accordance with the WS-Security 1.0 standard. A SAML token, included in the SOAP message, is used in SAML-based authentication with sender vouches confirmation. This policy uses WS-Security’s Basic 128 suite of asymmetric key technologies and SHA-1 hashing algorithm for message integrity. When this policy is selected, the recipient key alias <em>(recptAlias)</em> must be disabled.</td>
</tr>
<tr>
<td></td>
<td>- <strong>SAML_TOKEN_WITH_MSG_PROTECTION</strong> (oracle/wss10_saml_token_with_message_protection_client_policy)—This policy provides message-level protection (integrity and confidentiality) and SAML-based authentication for outbound SOAP requests in accordance with the WS-Security 1.0 standard. The Web service consumer includes a SAML token in the SOAP header and the confirmation type is sender-vouches. This policy uses WS-Security’s Basic 128 suite of asymmetric key technologies. Specifically, RSA key mechanisms for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption. and SHA-1 hashing algorithm for message integrity.</td>
</tr>
</tbody>
</table>
Portlet Producers

Oracle WebCenter Custom WLST Commands

5-73

tokenType

- **WSS11_SAML_TOKEN_WITH_MSG_PROTECTION** (oracle/wss11_saml_token_with_message_protection_client_policy)—This policy provides message-level protection (integrity and confidentiality) and SAML token population for outbound SOAP requests in accordance with the WS-Security 1.1 standard. A SAML token, included in the SOAP message, is used in SAML-based authentication with sender vouches confirmation. This policy uses the symmetric key technology for signing and encryption, and WS-Security’s Basic 128 suite of asymmetric key technologies for endorsing signatures.

- **WSS10_SAML_TOKEN_ONLY** (oracle/wss10_saml_token_client_policy)—This policy provides SAML-based authentication for outbound SOAP request messages in accordance with the WS-Security 1.0 standard. The policy propagates user identity and is typically used in intra departmental deployments where message protection and integrity checks are not required.

This policy does not require any keystore configuration.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>issuer</td>
<td>Optional. Name of the issuer of the token. The issuer name is the entity that vouches for the verification of the subject. For example: <a href="http://www.oracle.com">www.oracle.com</a>. This argument only applies when the tokenType is: SAML_TOKEN_WITH_MSG_PROTECTION, SAML_TOKEN_WITH_MSG_INTEGRITY, WSS10_SAML_TOKEN_ONLY, WSS11_SAML_TOKEN_WITH_MSG_PROTECTION.</td>
</tr>
<tr>
<td>defUser</td>
<td>Optional. User name to assert to the remote producer when the user is not authenticated with the WebCenter application. When unauthenticated, the identity anonymous is associated with the application user. The value anonymous may be inappropriate for the remote producer, so you may need to specify an alternative identity here. Keep in mind though, that in this case, the WebCenter application has not authenticated the user so the default user you specify should be a low privileged user in the remote producer. If the user has authenticated to the application, the user’s identity is asserted rather than the default user. This argument only applies when the tokenType is: USERNAME_WITHOUT_PASSWORD, SAML_TOKEN_WITH_MSG_PROTECTION, SAML_TOKEN_WITH_MSG_INTEGRITY, WSS10_SAML_TOKEN_ONLY, WSS11_SAML_TOKEN_WITH_MSG_PROTECTION.</td>
</tr>
<tr>
<td>extApp</td>
<td>Optional. This argument applies when the tokenType is USERNAME_WITHOUT_PASSWORD. If this producer uses an external application to store and supply user credentials for authentication, use this argument to name the associated external application.</td>
</tr>
<tr>
<td>keyStorePath</td>
<td>Optional. Full path to the key store that contains the certificate and the private key that is used for signing some parts of the SOAP message, such as the security token and SOAP message body. The selected file should be a key store created with the Java keytool.</td>
</tr>
<tr>
<td>keyStorePswwd</td>
<td>Optional. Password to the key store that was set when the key store was created.</td>
</tr>
<tr>
<td>sigKeyAlias</td>
<td>Optional. Identifier for the certificate associated with the private key that is used for signing.</td>
</tr>
<tr>
<td>sigKeyPswwd</td>
<td>Optional. Password for accessing the key identified by the alias that is specified using the sigKeyAlias argument.</td>
</tr>
<tr>
<td>encKeyAlias</td>
<td>Optional. Key alias to be used for encryption. A valid value is one of the key aliases that is located in the specified key store.</td>
</tr>
<tr>
<td>encKeyPswwd</td>
<td>Optional. Password for accessing the encryption key.</td>
</tr>
</tbody>
</table>
5.9.1.3 Examples

The following example registers a WSRP producer named WSRPSamples and registers the WSRP producer with an application named webcenter.

```
wlsc:/weblogic/serverConfig> registerWSRPProducer(appName='webcenter', name='WSRPSamples', url='http://myhost.com:9999/portletapp/portlets/wsrp2?WSDL')
```

The following example registers a secure WSRP producer.

```
wlsc:/weblogic/serverConfig> registerWSRPProducer(appName='webcenter', name='WSRPSamples2', url='http://myhost.com:8899/portletapp/portlets/wsrp2?WSDL', tokenType='WSS11_SAML_TOKEN_WITH_MSG_PROTECTION', issuer='www.oracle.com', defUser='anonymous', keyStorePath='/keys/mykeystore.jks', keyStorePswd='Test1', sigKeyAlias='mysigalias', sigKeyPswd='mysigpswd', encKeyAlias='myencalias', encKeyPswd='myencpswd', recptAlias='myrcptalias')
```

5.9.2 setWSRPProducer

Module: Oracle WebCenter

Use with WLST: Online

5.9.2.1 Description

Edits registration details for an existing WSRP producer.

5.9.2.2 Syntax

```
setWSRPProducer(appName, name, [url], [proxyHost], [proxyPort], [timeout], [externalApp], [tokenType],[issuer], [defUser], [keyStorePath], [keyStorePswd] [sigKeyAlias], [sigKeyPswd], [encKeyAlias], [encKeyPswd], [recptAlias], [server], [applicationVersion])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>name</td>
<td>Name of an existing WSRP producer.</td>
</tr>
</tbody>
</table>
### url
Optional. WSRP producer URL. The syntax will vary according to your WSRP implementation, for example:

- `http://host_name:port_number/context_root/portlets/wsrp2?WSDL`
- `http://host_name:port_number/context_root/portlets/wsrp1?WSDL`
- `http://host_name:port_number/context_root/portlets/?WSDL` (WSRP 1.0 for backward compatibility)

Where:
- `host_name` is the server where your producer is deployed
- `port_number` is the HTTP listener port number
- `context_root` is the Web application's context root
- `portlets[/wsrp(1|2)?WSDL]` is static text. The text entered here depends on how the producer is deployed.

For example:

```
```

### proxyHost
Optional. Host name or IP address of the proxy server.

A proxy is required when the WebCenter application and the remote portlet producer are separated by a firewall and an HTTP proxy is needed to communicate with the producer.

### proxyPort
Optional. Port number on which the proxy server listens.

### timeout
Optional. Timeout setting for communications with the producer, in seconds. For example, the maximum time the producer may take to register, deregister, or display portlets on WebCenter pages.

This argument defaults to 30.

Individual portlets may define their own timeout period, which takes precedence over the value expressed here.

### extApp
Optional. This argument applies when the `tokenType` is `USERNAME_ WITH_PASSWORD`. If this producer uses an external application to store and supply user credentials for authentication, use this argument to name the associated external application.
**tokenType**

Optional. Type of token profile to use for authentication with this WSRP producer. Valid values are:

- **USERNAME WITHOUT PASSWORD**
  (oracle/wss10_username_id_propagation_with_msg_protection_client_policy)—This policy provides message protection (integrity and confidentiality) and identity propagation for outbound SOAP requests in accordance with the WS-Security 1.0 standard. Credentials (username only) are included in outbound SOAP request messages through a WS-Security UsernameToken header. No password is included.

  Message protection is provided using WS-Security 1.0's Basic 128 suite of asymmetric key technologies. Specifically, RSA key mechanisms for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption.

- **USERNAME WITH PASSWORD**
  (oracle/wss10_username_token_with_message_protection_client_policy)—This policy provides message protection (integrity and confidentiality) and authentication for outbound SOAP requests in accordance with the WS-Security v1.0 standard. Both plain text and digest mechanisms are supported.

  This policy uses WS-Security's Basic 128 suite of asymmetric key technologies. Specifically, RSA key mechanism for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption.

  Use this token profile if the WSRP producer has a different identity store. You will need to define an external application pertaining to the producer and associate the external application with this producer.

- **SAML_TOKEN WITH MSG_INTEGRITY**
  (wss10_saml_token_with_message_integrity_client_policy)—This policy provides message-level integrity and SAML-based authentication for outbound SOAP requests in accordance with the WS-Security 1.0 standard. A SAML token, included in the SOAP message, is used in SAML-based authentication with sender vouches confirmation.

  This policy uses WS-Security's Basic 128 suite of asymmetric key technologies and SHA-1 hashing algorithm for message integrity.

  When this policy is selected, the recipient key alias (recptAlias) must be disabled.

- **SAML_TOKEN WITH MSG_PROTECTION**
  (oracle/wss10_saml_token_with_message_protection_client_policy)—This policy provides message-level protection (integrity and confidentiality) and SAML-based authentication for outbound SOAP requests in accordance with the WS-Security 1.0 standard. The Web service consumer includes a SAML token in the SOAP header and the confirmation type is sender-vouches.

  This policy uses WS-Security's Basic 128 suite of asymmetric key technologies. Specifically, RSA key mechanisms for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption. SHA-1 hashing algorithm for message integrity.

---

**Argument Definition**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>tokenType</strong></td>
<td>Optional. Type of token profile to use for authentication with this WSRP producer. Valid values are:</td>
</tr>
<tr>
<td><strong>USERNAME WITHOUT PASSWORD</strong></td>
<td>(oracle/wss10_username_id_propagation_with_msg_protection_client_policy)—This policy provides message protection (integrity and confidentiality) and identity propagation for outbound SOAP requests in accordance with the WS-Security 1.0 standard. Credentials (username only) are included in outbound SOAP request messages through a WS-Security UsernameToken header. No password is included. Message protection is provided using WS-Security 1.0's Basic 128 suite of asymmetric key technologies. Specifically, RSA key mechanisms for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption.</td>
</tr>
<tr>
<td><strong>USERNAME WITH PASSWORD</strong></td>
<td>(oracle/wss10_username_token_with_message_protection_client_policy)—This policy provides message protection (integrity and confidentiality) and authentication for outbound SOAP requests in accordance with the WS-Security v1.0 standard. Both plain text and digest mechanisms are supported. This policy uses WS-Security's Basic 128 suite of asymmetric key technologies. Specifically, RSA key mechanism for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption. Use this token profile if the WSRP producer has a different identity store. You will need to define an external application pertaining to the producer and associate the external application with this producer.</td>
</tr>
<tr>
<td><strong>SAML_TOKEN WITH MSG_INTEGRITY</strong></td>
<td>(wss10_saml_token_with_message_integrity_client_policy)—This policy provides message-level integrity and SAML-based authentication for outbound SOAP requests in accordance with the WS-Security 1.0 standard. A SAML token, included in the SOAP message, is used in SAML-based authentication with sender vouches confirmation. This policy uses WS-Security's Basic 128 suite of asymmetric key technologies and SHA-1 hashing algorithm for message integrity. When this policy is selected, the recipient key alias (recptAlias) must be disabled.</td>
</tr>
<tr>
<td><strong>SAML_TOKEN WITH MSG_PROTECTION</strong></td>
<td>(oracle/wss10_saml_token_with_message_protection_client_policy)—This policy provides message-level protection (integrity and confidentiality) and SAML-based authentication for outbound SOAP requests in accordance with the WS-Security 1.0 standard. The Web service consumer includes a SAML token in the SOAP header and the confirmation type is sender-vouches. This policy uses WS-Security's Basic 128 suite of asymmetric key technologies. Specifically, RSA key mechanisms for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption. SHA-1 hashing algorithm for message integrity.</td>
</tr>
<tr>
<td>Argument</td>
<td>Definition</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>tokenType</strong></td>
<td>- <strong>WSS11_SAML_TOKEN_WITH_MSG_PROTECTION</strong> (oracle/wss11_saml_token_with_message_protection_client_policy)—This policy enables message-level protection (integrity and confidentiality) and SAML token population for outbound SOAP requests in accordance with the WS-Security 1.1 standard. A SAML token, included in the SOAP message, is used in SAML-based authentication with sender vouches verification. This policy uses the symmetric key technology for signing and encryption, and WS-Security’s Basic 128 suite of asymmetric key technologies for endorsing signatures.</td>
</tr>
<tr>
<td></td>
<td>- <strong>WSS10_SAML_TOKEN_ONLY</strong> (oracle/wss10_saml_token_client_policy)—This policy includes SAML-tokens in outbound SOAP request messages in accordance with the WS-Security 1.0 standard. The policy propagates user identity and is typically used in intra departmental deployments where message protection and integrity checks are not required. This policy does not require any keystore configuration.</td>
</tr>
<tr>
<td><strong>issuer</strong></td>
<td>Optional. Name of the issuer of the token. The issuer name is the entity that vouches for the verification of the subject. For example: <a href="http://www.oracle.com">www.oracle.com</a>. This argument only applies when the tokenType is: SAML_TOKEN_WITH_MSG_PROTECTION, SAML_TOKEN_WITH_MSG_INTEGRITY, WSS10_SAML_TOKEN_ONLY, WSS11_SAML_TOKEN_WITH_MSG_PROTECTION.</td>
</tr>
<tr>
<td><strong>defUser</strong></td>
<td>Optional. User name to assert to the remote producer when the user is not authenticated with the WebCenter application. When unauthenticated, the identity anonymous is associated with the application user. The value anonymous may be inappropriate for the remote producer, so you may need to specify an alternative identity here. Keep in mind though, that in this case, the WebCenter application has not authenticated the user so the default user you specify should be a low privileged user in the remote producer. If the user has authenticated to the application, the user’s identity is asserted rather than the default user. This argument only applies when the tokenType is: USERNAME_WITHOUT_PASSWORD, SAML_TOKEN_WITH_MSG_PROTECTION, SAML_TOKEN_WITH_MSG_INTEGRITY, WSS10_SAML_TOKEN_ONLY, WSS11_SAML_TOKEN_WITH_MSG_PROTECTION.</td>
</tr>
<tr>
<td><strong>keyStorePath</strong></td>
<td>Optional. Full path to the key store that contains the certificate and the private key that is used for signing some parts of the SOAP message, such as the security token and SOAP message body. The selected file should be a key store created with the Java keytool.</td>
</tr>
<tr>
<td><strong>keyStorePswd</strong></td>
<td>Optional. Password to the key store that was set when the key store was created.</td>
</tr>
<tr>
<td><strong>sigKeyAlias</strong></td>
<td>Optional. Identifier for the certificate associated with the private key that is used for signing.</td>
</tr>
<tr>
<td><strong>sigKeyPswd</strong></td>
<td>Optional. Password for accessing the key identified by the alias that is specified using the sigKeyAlias argument.</td>
</tr>
<tr>
<td><strong>encKeyAlias</strong></td>
<td>Optional. Key alias used by the producer to encrypt the return message. A valid value is one of the key aliases that is located in the specified key store. If not specified, the producer uses the signing key for encrypting the return message.</td>
</tr>
<tr>
<td><strong>encKeyPswd</strong></td>
<td>Optional. Password for accessing the encryption key.</td>
</tr>
</tbody>
</table>
This example increases the timeout, for the WSRPSamples producer, to 60 seconds.

```
setWSRPProducer(appName='webcenter', name='WSRPSamples', timeout=60)
```

This example updates security properties on a secure WSRP producer.

```
setWSRPProducer(appName='webcenter', name='WSRPSamples2',
    tokenType='WSS11_SAML_TOKEN_WITH_MSG_PROTECTION',
    issuer='www.oracle.com', defUser='anonymous',
    keyStorePath='/keys/mykeystore.jks', keyStorePswd='Test1',
    sigKeyAlias='mysigalias', sigKeyPswd='mysigpswd',
    encKeyAlias='myencalias', encKeyPswd='myencpswd',
    recptAlias='myrcptalias')
```

This example removes all the security properties set on a secure WSRP producer.

```
setWSRPProducer(appName='webcenter', name='WSRPSamples2', tokenType='')
```

### 5.9.3 listWSRPProducers

**Module:** Oracle WebCenter

**Use with WLST:** Online

#### 5.9.3.1 Description

Lists WSRP producer registration details.

#### 5.9.3.2 Syntax

```
listWSRPProducers(appName,[name], [verbose], [server], [applicationVersion])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>The name of the application in which one or more WSRP producers is registered.</td>
</tr>
<tr>
<td>name</td>
<td>Optional. Name of an existing WSRP producer. If omitted, connection details for all WSRP producers configured for this WebCenter application are listed.</td>
</tr>
</tbody>
</table>
5.9.3.3 Example

```wls:/weblogic/serverConfig> listWSRPProducers(appName='webcenter', name='WSRPSamples')```

5.9.4 deregisterWSRPProducer

Module: Oracle WebCenter

Use with WLST: Online

5.9.4.1 Description

Deregisters a WSRP producer, and deletes the associated WSRP and Web Service connections.

5.9.4.2 Syntax

deregisterWSRPProducer(appName, name, [server], [applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application where the producer is registered.</td>
</tr>
<tr>
<td>name</td>
<td>Name of an existing WSRP producer.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.9.4.3 Example

The following example deregisters the WSRPSamples producer in an application named webcenter.

```wls:/weblogic/serverConfig> deregisterWSRPProducer(appName='webcenter', name='WSRPSamples')```
5.9.5 listWSRPProducerRegistrationProperties

Module: Oracle WebCenter

Use with WLST: Online

5.9.5.1 Description
Lists registration properties supported by a WSRP portlet producer.

5.9.5.2 Syntax
listWSRPProducerRegistrationProperties(appName, url,[proxyHost, [proxyPort],
[server], [applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>url</td>
<td>WSRP producer URL. The syntax will vary according to your WSRP implementation, for example:</td>
</tr>
<tr>
<td></td>
<td>http://host_name:port_number/context_root/portlets/wsrp2?WSDL</td>
</tr>
<tr>
<td></td>
<td>http://host_name:port_number/context_root/portlets/wsrp1?WSDL</td>
</tr>
<tr>
<td></td>
<td>http://host_name:port_number/context_root/portlets/?[WSRP 1.0 for backward compatibility)</td>
</tr>
<tr>
<td></td>
<td>Where:</td>
</tr>
<tr>
<td></td>
<td>■ host_name is the server where your producer is deployed</td>
</tr>
<tr>
<td></td>
<td>■ port_number is the HTTP listener port number</td>
</tr>
<tr>
<td></td>
<td>■ context_root is the Web application’s context root</td>
</tr>
<tr>
<td></td>
<td>■ portlets[/wsrp(1</td>
</tr>
<tr>
<td></td>
<td>For example:</td>
</tr>
<tr>
<td>proxyHost</td>
<td>Optional. Host name or IP address of the proxy server.</td>
</tr>
<tr>
<td>proxyPort</td>
<td>Optional. Port number on which the proxy server listens.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces.</td>
</tr>
<tr>
<td></td>
<td>Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.9.5.3 Example
The following example lists valid registration properties for the WSRP producer with the WSDL URL provided.

wls:/weblogic/serverConfig> listWSRPProducerRegistrationProperties
  (appName='webcenter', url='http://myhost:9999/portletapp/portlets/wsrp2?WSDL')
Registration Property hint : hint text
Registration Property label : label text
Registration Property language : en
Registration Property name : (urn:xyz:wlp:prop:reg:registration)consumerRole
Registration Property value : None

5.9.6 listWSRPPProducerUserCategories

Module: Oracle WebCenter
Use with WLST: Online

5.9.6.1 Description
Lists any user categories that a WSRP producer might support. WebCenter users can use the WLST command mapWSRPPProducerUserCategory to map application roles to a producer's user category.

5.9.6.2 Syntax
listWSRPPProducerUserCategories(appName, name, [server], [applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>name</td>
<td>Name of an existing WSRP producer.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.9.6.3 Example
The following example displays the categories associated with a WSRP producer named WSRPSamples.

wls:/weblogic/serverConfig> listWSRPPProducerUserCategories(appName='webcenter', name='WSRPSamples')
User Category Name : categoryTwo
User Category Description : Custom role two.
User Category Mapped Local Roles : None

User Category Name : categoryOne
User Category Description : Custom role one.
User Category Mapped Local Roles : None

5.9.7 mapWSRPPProducerUserCategory

Module: Oracle WebCenter
Use with WLST: Online
5.9.7.1 Description
Maps a role that is defined in the specified WebCenter application to a user category supported by a WSRP producer. The user categories may be found using listWSRPProducerUserCategories.

5.9.7.2 Syntax
mapWSRPProducerUserCategory(appName, name, localRole, producerUserCategory, [server], [applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>name</td>
<td>Name of an existing WSRP producer.</td>
</tr>
<tr>
<td>localRole</td>
<td>Name of the WebCenter application role to be mapped.</td>
</tr>
<tr>
<td>producerUserCategory</td>
<td>WSRP producer user category to which the WebCenter role will be mapped.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.9.7.3 Example
The following example maps the application role admin to the WSRP user category wrsp-admin.

wls:/weblogic/serverConfig> mapWSRPProducerUserCategory(appName='webcenter', name='WSRPProducer1', localRole='admin', producerUserCategory='wsrp-admin')

5.9.8 registerPDKJavaProducer
Module: Oracle WebCenter
Use with WLST: Online

5.9.8.1 Description
Creates a connection to an Oracle PDK-Java portlet producer and registers the Oracle PDK-Java producer with a named WebCenter application.

5.9.8.2 Syntax
registerPDKJavaProducer(appName, name, url, [serviceId], [proxyHost, [proxyPort]], [subscriberId], [sharedKey], [timeout], [establishSession], [externalApp], [mapUser], [server], [applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application for which you want to perform this operation.</td>
</tr>
<tr>
<td>name</td>
<td>Connection name. The name must be unique (across all connection types) within the WebCenter application.</td>
</tr>
<tr>
<td>Argument</td>
<td>Definition</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| url         | URL for the Oracle PDK-Java producer. Use the following syntax: http://host_name:port_number/context_root/providers  
Where:  
- host_name is the server where the producer is deployed  
- port_number is the HTTP Listener port number  
- context_root is the Web application's context root.  
- providers is static text. The text entered here depends on how the producer is deployed.  
For example: http://myHost:7778/myEnterprisePortlets/providers |
| serviceId   | Optional. Service ID of the producer.  
PDK-Java enables you to deploy multiple producers under a single adapter servlet. Producers are identified by their unique service ID. A service ID is required only if the service ID is not appended to the URL end point.  
For example, the following URL endpoint requires sample as the service ID: http://domain.us.oracle.com:7778/axyz/providers  
However, the following URL endpoint, does not require a service ID: http://domain.us.oracle.com:7778/axyz/providers/sample  
The service ID is used to look up a file called <service_id>.properties, which defines the characteristics of the producer, such as whether to display its test page. Use any value to create the service ID. |
| proxyHost   | Optional. Host name or IP address of the proxy server.  
A proxy is required if the WebCenter application and the remote portlet producer are separated by a firewall and an HTTP proxy is needed for communication with the producer. |
| proxyPort   | Optional. Port number on which the proxy server listens. This argument defaults to 80.                                                                                                                      |
| sharedKey   | Optional. Shared key used for message authentication with the remote producer. Message authentication ensures that the incoming messages are sent from a host with a shared key. This argument defaults to null.  
The shared key can contain between 10 and 20 alphanumeric characters. |
| subscriberId| Optional. Consumer's identifier, if required.  
When a producer is registered with an application, a call is made to the producer. During the call, the consumer (WebCenter application in this instance) passes the value for subscriberId to the producer. The producer may be coded to use the subscriber ID. |
| timeout     | Optional. Timeout setting for communications with the producer, in seconds. For example, the maximum time the producer may take to register, deregister, or display portlets on WebCenter pages.  
This argument defaults to 30.  
Individual portlets may define their own timeout period, which takes precedence over the value expressed here. |
5.9.8.3 Example

The following example creates and registers an Oracle PDK-Java producer named JPDKSamples, for an application named webcenter.

```
wls:/weblogic/serverConfig> registerPDKJavaProducer(appName='webcenter', name='JPDKSamples', url='http://myhost:9999/jpdk/providers/sample')
```

5.9.9 setPDKJavaProducer

Module: Oracle WebCenter

Use with WLST: Online

5.9.9.1 Description

Edits registration details for an existing PDK-Java producer.

5.9.9.2 Syntax

```
snPDKJavaProducer(appName, name, url, [serviceId], [proxyHost, [proxyPort]], [subscriberId], [sharedKey], [timeout], [establishSession], [externalApp], [mapUser], [server], [applicationVersion])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>name</td>
<td>Name of an existing PDK-Java producer.</td>
</tr>
<tr>
<td>Argument</td>
<td>Definition</td>
</tr>
<tr>
<td>-----------</td>
<td>------------</td>
</tr>
</tbody>
</table>
| url       | URL for the Oracle PDK-Java producer. Use the following syntax:  
                        | http://host_name:port_number/context_root/providers  
                        Where:  
                        ■ host_name is the server where the producer is deployed  
                        ■ port_number is the HTTP Listener port number  
                        ■ context_root is the Web application’s context root.  
                        ■ providers is static text. The text entered here depends on how  
                        the producer is deployed.  
                        For example:  
                        http://myHost:7778/myEnterprisePortlets/providers  
| serviceId | Optional. Service ID of the producer.  
                        PDK-Java enables you to deploy multiple producers under a single  
                        adapter servlet. Producers are identified by their unique service ID. A  
                        service ID is required only if the service ID is not appended to the  
                        URL end point.  
                        For example the following URL endpoint requires sample as the  
                        service ID:  
                        http://domain.us.oracle.com:7778/axyz/providers  
                        However, the following URL endpoint, does not require a service ID:  
                        http://domain.us.oracle.com:7778/axyz/providers/sample  
                        The service ID is used to look up a file called <service_  
                        id>.properties, which defines the characteristics of the producer,  
                        such as whether to display its test page. Use any value to create the  
                        service ID.  
| proxyHost | Optional. Host name or IP address of the proxy server.  
                        A proxy is required if the WebCenter application and the remote  
                        portlet producer are separated by a firewall and an HTTP proxy is  
                        needed for communication with the producer.  
| proxyPort | Optional. Port number on which the proxy server listens.  
| subscriberId | Optional. Consumer’s identifier, if required.  
                        When a producer is registered with an application, a call is made to  
                        the producer. During the call, the consumer (WebCenter application  
                        in this instance) passes the value for Subscriber ID to the producer. If  
                        the producer does not see the expected value for Subscriber ID, it  
                        might reject the registration call.  
| sharedKey | Optional. The shared key is used for message authentication with the  
                        remote producer. Message authentication ensures that the incoming  
                        messages are sent from a host with a shared key. You should enable  
                        sessions using the sharedKey argument, as well as the  
                        establishSession argument.  
| timeout   | Optional. Timeout setting for communications with the producer, in  
                        seconds. For example, the maximum time the producer may take to  
                        register, deregister, or display portlets on WebCenter pages.  
                        Individual portlets may define their own timeout period, which takes  
                        precedence over the value expressed here.  

Portlet Producers

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>establishSession</td>
<td>Optional. Enable a user session when executing portlets from this producer. Valid values are true and false. You should enable sessions using the establishSession argument, as well as the sharedKey argument. When sessions are enabled (true), the server maintains session-specific information, such as the user name. Message authentication uses sessions, so if a shared key is specified, this option should also be enabled. For sessionless communication between the producer and the server, set to false.</td>
</tr>
<tr>
<td>externalApp</td>
<td>Optional. Name of the external application associated with this producer.</td>
</tr>
<tr>
<td>mapUser</td>
<td>Optional. Flag indicating whether the mapped user name from the external application should be passed to the producer. Valid values are true and false.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

### 5.9.10 deregisterPDKJavaProducer

Module: Oracle WebCenter
Use with WLST: Online

#### 5.9.10.1 Description
Deregisters an Oracle PDK-Java producer and deletes the associated connection, for a named WebCenter application.

#### 5.9.10.2 Syntax
deregisterPDKJavaProducer(appName, name, [server, applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>name</td>
<td>Name of an existing PDK-Java producer.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
</tbody>
</table>
Portlet Producers

5.9.10.3 Example
The following example deregisters the wc-WebClipping producer, and deletes the associated connection.

wls:/weblogic/serverConfig> deregisterPDKJavaProducer(appName='webcenter', name='wc-WebClipping')
Already in Domain Runtime Tree
Producer wc-WebClipping has been deregistered.
Already in Domain Runtime Tree
'wc-WebClipping' successfully deleted
Already in Domain Runtime Tree
'wc-WebClipping-urlconn' successfully deleted

5.9.11 listPDKJavaProducers
Module: Oracle WebCenter
Use with WLST: Online

5.9.11.1 Description
Lists details for one or more Oracle PDK-Java producers registered with a named WebCenter application.

5.9.11.2 Syntax
listPDKJavaProducers(appName, [name],[verbose], [server, applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>name</td>
<td>Optional. Name of an existing PDK-Java portlet producer. If omitted, connection details for all PDK-Java producers configured for this WebCenter application are listed.</td>
</tr>
<tr>
<td>verbose</td>
<td>Optional. Displays PDK-Java producer connection details in verbose mode. Valid options are true and false. When set to true, listPDKJavaProducers lists all connection properties. When set to false, listPDKJavaProducers lists connection names only. This argument defaults to true. If you set this argument to false, do not specify the name argument.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.9.11.3 Example
The following example lists all the connection properties (verbose mode) for the JPDKSamples producer.
Portlet Producers

5.9.12 refreshProducer
Module: Oracle WebCenter
Use with WLST: Online

5.9.12.1 Description
Refreshes the metadata stored for a named producer to reflect the portlets that are currently offered by that producer.

5.9.12.2 Syntax
refreshProducer(appName, producerName, [server, applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which the producer is registered.</td>
</tr>
<tr>
<td>producerName</td>
<td>Name of an existing producer.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.9.12.3 Example
The following example refreshes the WSRPSamples producer in an application named webcenter.

wls:/weblogic/serverConfig> refreshProducer(appName='webcenter', producerName='WSRPSamples')
Producer WSRPSamples has been refreshed.

5.9.13 registerOOTBProducers
Module: Oracle WebCenter
Use with WLST: Online

5.9.13.1 Description

5.9.13.2 Syntax
registerOOTBProducers(producerHost, producerPort, appName, [server, applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>producerHost</td>
<td>Host name or IP address of the server hosting out-of-the-box producers.</td>
</tr>
<tr>
<td>producerPort</td>
<td>Port number for the server hosting out-of-the-box producers.</td>
</tr>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which the out-of-the-box producers are to be registered.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.9.13.3 Example
The following example registers out-of-the-box producers in a WebCenter application named myApp.

```
wls:/weblogic/serverConfig> registerOOTBProducers(producerHost='myhost.com', producerPort=9999, appName='myApp')
```

Registering Out-of-the-Box Producers
Registering producers at http://myhost.com:9999

Registering OmniPortlet
Created connection wc-OmniPortlet-urlconn
Created connection wc-OmniPortlet
Producer connection wc-OmniPortlet has been registered.

Registering WebClipping
Created connection wc-WebClipping-urlconn
Created connection wc-WebClipping
Producer connection wc-WebClipping has been registered.

Registering RichTextPortlet
Created connection wc-RichText-wsconn
Created connection wc-RichText
Producer connection wc-RichText has been registered.

Registering WSRP Tools
Created connection wc-WSRPTools-wsconn
Created connection wc-WSRPTools
Producer connection wc-WSRPTools has been registered.
5.9.14 deregisterOOTBProducers

Module: Oracle WebCenter

Use with WLST: Online

5.9.14.1 Description

5.9.14.2 Syntax
```
deregisterOOTBProducers(appName, [server, applicationVersion])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which the out-of-the-box producers are currently registered.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.9.14.3 Example
The following example deregisters out-of-the-box WebCenter producers, and deletes their associated connections, in an application named myApp.

```
wls:/weblogic/serverConfig> deregisterOOTBProducers(appName='myApp')
```

Deregistering Out-of-the-Box Producers

Deregistering OmniPortlet
Producer wc-OmniPortlet has been deregistered.
wC-OmniPortlet successfully deleted
wc-OmniPortlet-urlconn successfully deleted

Deregistering WebClipping
Producer wc-WebClipping has been deregistered.
wC-WebClipping successfully deleted
wc-WebClipping-urlconn successfully deleted

Deregistering RichTextPortlet
Producer wc-RichText has been deregistered.
wC-RichText successfully deleted
wc-RichText-wsconn successfully deleted

Deregistering WSRP Tools
Producer wc-WSRPTools has been deregistered.
wC-WSRPTools successfully deleted
wc-WSRPTools-wsconn successfully deleted

5.9.15 registerSampleProducers

Module: Oracle WebCenter

Use with WLST: Online
5.9.15.1 Description
Registers the sample producers provided with Oracle WebCenter with a named WebCenter application. There are two sample producers — WSRP Samples and JPDK Samples.

5.9.15.2 Syntax
registerSampleProducers(producerHost, producerPort, appName, [server, applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>producerHost</td>
<td>Host name or IP address of the server hosting the sample producers.</td>
</tr>
<tr>
<td>producerPort</td>
<td>Port number for the server hosting the sample producers.</td>
</tr>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which the sample producers are to be registered.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.9.15.3 Example
The following example registers Oracle WebCenter sample producers in an application named myApp.

wls:/weblogic/serverConfig> registerSampleProducers(producerHost='myhost.com', producerPort=9999, appName='myApp')

5.9.16 deregisterSampleProducers
Module: Oracle WebCenter
Use with WLST: Online

5.9.16.1 Description
Deregisters the Oracle WebCenter sample producers (WSRP Samples and JPDK Samples) from a named WebCenter application.

5.9.16.2 Syntax
deregisterSampleProducers(appName, [server, applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which the sample producers are currently registered. If a value is not specified, this argument defaults to webcenter.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>
5.9.16.3 Example
The following example deregisters sample producers from a WebCenter application named myApp.

```
wls:/weblogic/serverConfig> deregisterSampleProducers(appName='myApp')
```

5.10 RSS

Use the commands listed in Table 5–17 to manage proxy settings for the RSS service. Configuration changes made using these WebCenter WLST commands are only effective after your restart the Managed Server on which the WebCenter application is deployed. For details, see Oracle Fusion Middleware Administrator’s Guide for Oracle WebCenter.

<table>
<thead>
<tr>
<th>Table 5–17 RSS WLST Commands</th>
</tr>
</thead>
</table>

### 5.10.1 getRssProxyConfig
Module: Oracle WebCenter
Use with WLST: Online

5.10.1.1 Description
Returns the proxy host and proxy port used by the RSS service. Depending on your network configuration, proxy details may be required to display external RSS news feeds in your WebCenter application.

5.10.1.2 Syntax
```
getRssProxyConfig(appName, [server, applicationVersion])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.10.1.3 Example
The following example returns the proxy host and proxy port used by the RSS service in a WebCenter application named webcenter.

```
wls:/weblogic/serverConfig> getRssProxyConfig(appName='webcenter')
```
5.10.2 setRssProxyConfig

Module: Oracle WebCenter
Use with WLST: Online

5.10.2.1 Description
Specifies the proxy host and port for the RSS service. Depending on your network configuration, proxy details may be required to display external RSS news feeds in your WebCenter application.

5.10.2.2 Syntax
setRssProxyConfig(appName, proxyHost, proxyPort, [server, applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>proxyHost</td>
<td>Host name of the proxy server.</td>
</tr>
<tr>
<td>proxyPort</td>
<td>Port on which the proxy server is running.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.10.2.3 Example
The following example sets the proxy host and proxy port used by the RSS service in a WebCenter application named webcenter.

wls:/weblogic/serverConfig> setRssProxyConfig(appName='webcenter', proxyHost='www-proxy.example.com', proxyPort='80')

5.11 Search

Use the commands listed in Table 5-18 to manage Oracle Secure Enterprise Search (SES) connections and other search related properties for WebCenter applications.

Configuration changes made using these WebCenter WLST commands are only effective after your restart the Managed Server on which the WebCenter application is deployed. For details, see Oracle Fusion Middleware Administrator’s Guide for Oracle WebCenter.

Table 5-18  Search WLST Commands

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>createSESConnection</td>
<td>Create a connection to an SES instance for a WebCenter application.</td>
<td>Online</td>
</tr>
<tr>
<td>setSESConnection</td>
<td>Edit an existing SES search connection.</td>
<td>Online</td>
</tr>
<tr>
<td>listSESConnections</td>
<td>List individual or all SES search connections that are configured for a specific WebCenter application.</td>
<td>Online</td>
</tr>
</tbody>
</table>
5.11.1 createSESConnection

Module: Oracle WebCenter
Use with WLST: Online

5.11.1.1 Description

Creates a connection to an Oracle Secure Enterprise Search (SES) instance for a WebCenter application.

5.11.1.2 Syntax

createSESConnection(appName, name, url, appUser, appPassword, [default], [server], [applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>name</td>
<td>Connection name. The name must be unique (across all connection types) within the WebCenter application.</td>
</tr>
<tr>
<td>url</td>
<td>Web Services URL that Oracle Secure Enterprise Search exposes to enable Search requests. Use the format: http://&lt;host&gt;:&lt;port&gt;/search/query/OracleSearch</td>
</tr>
<tr>
<td>appUser</td>
<td>User name that the WebCenter application uses to authenticate itself as a trusted application to Oracle Secure Enterprise Search so that it may perform searches on behalf of WebCenter users. The specified user must be present in both the Oracle Identity Management server configured for the WebCenter application and the Oracle Identity Management server configured for Oracle SES.</td>
</tr>
<tr>
<td>appPassword</td>
<td>Password for the user name specified.</td>
</tr>
<tr>
<td>default</td>
<td>Optional. Configures WebCenter Search service to actively use the search connection. Valid options are true and false. Setting to true replaces any other search connection that is being used. Setting to false does not change the current Search service configuration. This argument defaults to false.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>
5.11.3 Example
The following example creates a new search connection that points to the SES instance http://myhost.com:7777/search/query/OracleSearch and makes this connection the active search connection for the WebCenter application named app1.

```
wlst:/weblogic/serverConfig> createSESConnection(appName='app1', name='SESConn1', url='http://myhost.com:7777/search/query/OracleSearch', appUser='wpadmin', appPassword='password', default=true)
```

5.11.2 setSESConnection
Module: Oracle WebCenter
Use with WLST: Online

5.11.2.1 Description
Edits an existing Oracle Secure Enterprise Search (SES) search connection.

5.11.2.2 Syntax
```
setSESConnection(appName, name, [url], [appUser], [appPassword], [default], [server], [applicationVersion])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>name</td>
<td>Name of an existing search connection.</td>
</tr>
<tr>
<td>url</td>
<td>Optional. Web Services URL that Oracle Secure Enterprise Search exposes to enable Search requests. Use the format: http://&lt;host&gt;:&lt;port&gt;/search/query/OracleSearch</td>
</tr>
<tr>
<td>appUser</td>
<td>Optional. User name that the WebCenter application uses to log in to Oracle Secure Enterprise Search so that it may perform searches on behalf of WebCenter users.</td>
</tr>
<tr>
<td>appPassword</td>
<td>Optional. Password that the WebCenter application uses to log in to Oracle Secure Enterprise Search so that it may perform searches on behalf of WebCenter users.</td>
</tr>
<tr>
<td>default</td>
<td>Optional. Configures WebCenter Search service to actively use the search connection. Valid options are true and false. Setting to true replaces any other search connection that is being used. Setting to false does not change the current Search service configuration. This argument defaults to false.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.11.2.3 Example
The following example modifies the URL of a search connection named SESConn1 and makes the connection the active search connection for a WebCenter application named app1.

```
wlst:/weblogic/serverConfig> setSESConnection(appName='app1', name='SESConn1', url='http://myhost.com:7777/search/query/OracleSearch', appUser='wpadmin', appPassword='password')
```
5.11.3 listSESConnections

Module: Oracle WebCenter

Use with WLST: Online

5.11.3.1 Description
Lists the names of all Oracle Secure Enterprise Search (SES) search connections configured for a WebCenter application.

5.11.3.2 Syntax
listSESConnections(appName, [verbose], [name], [server], [applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application for which you want to perform this operation.</td>
</tr>
<tr>
<td>verbose</td>
<td>Optional. Displays search connection details in verbose mode. Valid options are true and false. When set to true, listSESConnections lists all of the search connections that are configured for a WebCenter application, along with their details. When set to false, listSESConnections lists connection names only. This argument defaults to false. If you set this argument to false, do not specify the name argument.</td>
</tr>
<tr>
<td>name</td>
<td>Optional. Name of an existing search connection. You can use this argument to view details about a specific connection.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.11.3.3 Examples
The following example displays connection details for all search connections configured for a WebCenter application, named WebCenterApp.

wls:/weblogic/serverConfig> listSESConnections(appName='WebCenterApp', verbose='true')

The following example displays connection details for a search connection named SESConn1.

wls:/weblogic/serverConfig> listSESConnections(appName='WebCenterApp', verbose='true', name='SESConn1')

5.11.4 setSearchConfig

Module: Oracle WebCenter

Use with WLST: Online
5.11.4.1 Description
Modifies search settings for a WebCenter application. If a parameter is not specified it is not modified.

5.11.4.2 Syntax
setSearchConfig(appName, [numSavedSearches], [numResultsRegion], [numResultsMain],
[numResultsToolbar], [executionTimeout], [prepareTimeout], [showAllExecutionTimeout],
[server], [applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>numSavedSearches</td>
<td>Optional. The number of saved searches to display in the Saved Searches drop down (on main search page).</td>
</tr>
<tr>
<td>numResultsRegion</td>
<td>Optional. The number of saved searches displayed in a Saved Search task flow.</td>
</tr>
<tr>
<td>numResultsMain</td>
<td>Optional. The number of search results displayed, per service, for searches submitted from the main search page.</td>
</tr>
<tr>
<td>numResultsToolbar</td>
<td>Optional. The number of search results displayed, per service, for searches submitted from the global search toolbar. The value for this argument must be a valid number.</td>
</tr>
<tr>
<td>executionTimeout</td>
<td>Optional. The maximum time that a service is allowed to execute a search (in ms). The value for this argument must be a valid number.</td>
</tr>
<tr>
<td>prepareTimeout</td>
<td>Optional. The maximum time that a service is allowed to initialize a search (in ms). The value for this argument must be a valid number.</td>
</tr>
<tr>
<td>showAllExecutionTimeout</td>
<td>Optional. The maximum time that a service is allowed to display search all results (in ms). The value for this argument must be a valid number.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.11.4.3 Examples
The following example specifies that saved searches display five search results per service. Additionally, that a seven second search execution timeout is required.

```wls:/weblogic/serverConfig> setSearchConfig(appName='webcenter',
numResultsRegion=5, executionTimeout=7000);```

The following example increases the number of saved searches in the Saved Searches drop down list to eight, while changing the number of results from global toolbar searches to ten, per service.

```wls:/weblogic/serverConfig> setSearchConfig(appName='webcenter',
numSavedSearches=8, numResultsToolbar=10);```

The following example sets the search execution timeout to five seconds and allows each service fifteen seconds to display search results before timing out.

```wls:/weblogic/serverConfig> setSearchConfig(appName='webcenter',
```
executionTimeout=5000, showAllExecutionTimeout=15000);

5.11.5 setSearchSESConfig

Module: Oracle WebCenter
Use with WLST: Online

5.11.5.1 Description
Configures search settings for an existing Oracle Secure Enterprise Search (SES) search connection. If a parameter is not specified it is not modified.

5.11.5.2 Syntax
setSearchSESConfig(appName, [connectionName], [dataGroup], [topNRows], [server], [applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>connectionName</td>
<td>Optional. Names the search connection that the Search service must use.</td>
</tr>
<tr>
<td>dataGroup</td>
<td>Optional. Names the Secure Enterprise Search data group in which to search. If a value is not provided, everything in the Oracle Secure Enterprise Search instance will be searched.</td>
</tr>
<tr>
<td>topNRows</td>
<td>Optional. Number of top N rows of search results for gathering refinement data.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.11.5.3 Example
The following example specifies that the Search service must use the search connection named SESConn1, and to search the data group named group2.

wls:/weblogic/serverConfig> setSearchSESConfig(appName='webcenter', connectionName='SESConn1', dataGroup='group2', topNRows=200);

The following example changes the maximum number of search results that the Search service returns. No connection name is specified, in this example, so this configuration change is applied to the current default (or active) search connection.

wls:/weblogic/serverConfig> setSearchSESConfig(appName='webcenter', topNRows=500);
Already in Domain Runtime Tree
Restart is needed for the service connection changes to take effect.

5.11.6 listSearchConfig

Module: Oracle WebCenter
Use with WLST: Online
5.11.6.1 Description
Lists search settings for a WebCenter application.

5.11.6.2 Syntax
listSearchConfig(appName, [server, applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application for which you want to perform this operation.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.11.6.3 Example
The following example displays search configuration information for a WebCenter application named webcenter.

wls:/weblogic/serverConfig> listSearchConfig(appName='webcenter')

5.11.7 listSearchSESConfig
Module: Oracle WebCenter
Use with WLST: Online

5.11.7.1 Description
Lists SES search settings for a WebCenter application.

5.11.7.2 Syntax
listSearchSESConfig(appName, [server, applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application for which you want to perform this operation.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.11.7.3 Example
The following example displays SES search configuration information for a WebCenter application named webcenter.

wls:/weblogic/serverConfig> listSearchSESConfig(appName='webcenter')
Already in Domain Runtime Tree
----------------
Search SES Config
5.12 WebCenter Spaces Crawlers

Use the commands listed in Table 5–19 to manage crawl settings for WebCenter Spaces.

Configuration changes made using these WebCenter WLST commands are effective immediately; no restart is required.

<table>
<thead>
<tr>
<th>Table 5–19</th>
<th>WebCenter Spaces Crawler WLST Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use This Command...</td>
<td>To...</td>
</tr>
<tr>
<td>setSpacesCrawlProperties</td>
<td>Specify crawl properties for WebCenter Spaces.</td>
</tr>
<tr>
<td>getSpacesCrawlProperties</td>
<td>Return the current crawl settings for WebCenter Spaces.</td>
</tr>
</tbody>
</table>

5.12.1 setSpacesCrawlProperties

Module: Oracle WebCenter

Use with WLST: Online

5.12.1.1 Description

Specifies crawl properties for WebCenter Spaces.

WebCenter Spaces can be crawled by Oracle SES to provide a faster, more unified search experience across WebCenter objects, specifically: group spaces, lists, pages, people (profiles), wikis, blogs, documents, discussions, and announcements. Three distinct crawlers make this possible:

- WebCenter Spaces Crawler (for group spaces, lists, pages, people, wikis, and blogs)
- Documents Crawler (for documents)
- Discussions Crawler (for discussions and announcements).

You can use this command to enable or disable these Oracle SES crawlers in WebCenter Spaces.

You can also use this command to specify an interval between full crawls for the WebCenter Spaces Crawler. During a full crawl, all of the WebCenter Spaces Crawler content is re-read. Out-of-the-box, full crawls for the WebCenter Spaces Crawler occur every seven days but you can specify a different frequency to suit your installation.

Note that incremental crawls, for all three crawlers, are initiated by a scheduler running from Oracle SES. During these incremental crawls, only content added or updated since the previous crawl is processed.

5.12.1.2 Syntax

`setSpacesCrawlProperties(appName, [fullCrawlIntervalInHours, spacesCrawlEnabled, documentCrawlEnabled, discussionsCrawlEnabled, server, applicationVersion])`
5.12.1.3 Example
The following example specifies that WebCenter Spaces runs a full crawl through the WebCenter Spaces Crawler every 8 days.

```
wls:/weblogic/serverConfig> setSpacesCrawlProperties(appName='webcenter', fullCrawlIntervalInHours=192)
```

5.12.2 getSpacesCrawlProperties

Module: Oracle WebCenter
Use with WLST: Online

5.12.2.1 Description
Returns the current crawl settings for WebCenter Spaces, such as the number of hours between full crawls (WebCenter Spaces crawler), and which Oracle SES crawlers are enabled (WebCenter Spaces crawler, Discussions crawler, and Document crawler).

5.12.2.2 Syntax
```
getSpacesCrawlProperties(appName, [server, applicationVersion])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter Spaces application—always webcenter.</td>
</tr>
<tr>
<td>fullCrawlIntervalInHours</td>
<td>Optional. Number of hours between full crawls. The default is 168 hours or 7 days.</td>
</tr>
<tr>
<td>spacesCrawlEnabled</td>
<td>Optional. Specifies whether the WebCenter Spaces Crawler is enabled in Oracle SES. Valid values are true and false. This argument defaults to false. When set to false, WebCenter's internal search adapters return search results.</td>
</tr>
<tr>
<td>documentCrawlEnabled</td>
<td>Optional. Specifies whether the Documents Crawler is enabled in Oracle SES. Valid values are true and false. This argument defaults to false. When set to false, WebCenter's internal search adapters return search results.</td>
</tr>
<tr>
<td>discussionsCrawlEnabled</td>
<td>Optional. Specifies whether the Discussions Crawler is enabled in Oracle SES. Valid values are true and false. This argument defaults to false. When set to false, WebCenter's internal search adapters return search results.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>
5.12.2.3 Example

The following example returns the current crawl settings for WebCenter Spaces.

```
wls:/weblogic/serverConfig> getSpacesCrawlProperties(appName='webcenter')
```

WebCenter Spaces Crawl Properties:
-----------------
fullCrawlIntervalInHours: 124
spacesCrawlEnabled: true
documentCrawlEnabled: true
discussionsCrawlEnabled: false

5.13 Worklists

Use the commands listed in Table 5–20 to manage BPEL server connections for WebCenter applications.

Configuration changes made using these WebCenter WLST commands are only effective after you restart the Managed Server on which the WebCenter application is deployed. For details, see Oracle Fusion Middleware Administrator’s Guide for Oracle WebCenter.

### Table 5–20 Worklist Commands

<table>
<thead>
<tr>
<th>Use this command…</th>
<th>To…</th>
<th>Use with WLST…</th>
</tr>
</thead>
<tbody>
<tr>
<td>createBPELConnection</td>
<td>Create a connection to a BPEL server for a WebCenter application.</td>
<td>Online</td>
</tr>
<tr>
<td>setBPELConnection</td>
<td>Edit an existing BPEL server connection.</td>
<td>Online</td>
</tr>
<tr>
<td>listBPELConnections</td>
<td>List all of the BPEL server connections that are configured for a WebCenter application.</td>
<td>Online</td>
</tr>
<tr>
<td>addWorklistConnection</td>
<td>Enable an existing BPEL server connection for the Worklist service.</td>
<td>Online</td>
</tr>
<tr>
<td>removeWorklistConnection</td>
<td>Disable a BPEL server connection currently used by the Worklist service.</td>
<td>Online</td>
</tr>
<tr>
<td>listWorklistConnections</td>
<td>List individual or all BPEL server connections configured for the Worklist service.</td>
<td>Online</td>
</tr>
</tbody>
</table>

5.13.1 createBPELConnection

Module: Oracle WebCenter

Use with WLST: Online
5.13.1.1 Description

Creates a connection to a BPEL server for a named WebCenter application. BPEL server connections can be used by the application's Worklist service and WebCenter Spaces workflows.

To configure the Worklist service to actively use a new BPEL server connection, use the `addWorklistConnection` command. See Section 5.13.4, "addWorklistConnection".

To specify the BPEL server connection that WebCenter Spaces uses for its internal workflows, use the `setSpacesWorkflowConnectionName` command. See Section 5.14.2, "setSpacesWorkflowConnectionName".

5.13.1.2 Syntax

```
createBPELConnection(appName, name, url, [policy], [server], [applicationVersion])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>appName</code></td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td><code>name</code></td>
<td>Connection name. The name must be unique (across all connection types) within the WebCenter application.</td>
</tr>
<tr>
<td><code>url</code></td>
<td>URL required to access the BPEL server. Use the format: <code>&lt;protocol&gt;://&lt;host&gt;:&lt;port&gt;</code></td>
</tr>
<tr>
<td><code>policy</code></td>
<td>Optional. SAML token policy this connection uses for authentication. This argument defaults to <code>oracle/wss10_saml_token_client_policy</code>.</td>
</tr>
<tr>
<td><code>server</code></td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, <code>WLS_Spaces</code>. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td><code>applicationVersion</code></td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.13.1.3 Examples

The following example creates a connection named `WebCenter Worklist` with the URL value provided:

```
wls:/weblogic/serverConfig> createBPELConnection(appName='webcenter', name='WebCenter Worklist', url='http://myhost.com:8001', policy='oracle/wss10_saml_token_client_policy')
```

The following example creates a connection that uses the default security policy:

```
wls:/weblogic/serverConfig> createBPELConnection(appName='webcenter', name='WebCenter Worklist', url='http://myhost.com:8001')
```

5.13.2 setBPELConnection

Module: Oracle WebCenter

Use with WLST: Online
5.13.2.1 Description
Edits an existing BPEL server connection. The BPEL server URL and the security
policy are both editable.

To configure the Worklist service to actively use an existing BPEL server connection,
use the addWorklistConnection command. See Section 5.13.4,
"addWorklistConnection".

To specify the BPEL server connection used for Webcenter Spaces workflows, use the
setSpacesWorkflowConnectionName command. See Section 5.14.2,
"setSpacesWorkflowConnectionName".

5.13.2.2 Syntax
setBPELConnection(appName, name,[url],[policy],[server], [applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>name</td>
<td>Existing BPEL server connection name.</td>
</tr>
<tr>
<td>url</td>
<td>Optional. URL required to access the BPEL server. Use the format: &lt;protocol&gt;://&lt;host&gt;:&lt;port&gt; The BPEL server URL must be unique within the WebCenter application.</td>
</tr>
<tr>
<td>policy</td>
<td>Optional. SAML token policy this connection uses for authentication. This argument defaults to oracle/wss10_saml_token_client_policy.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.13.2.3 Examples
The following example updates the BPEL server URL and security policy for a
connection named WebCenter Worklist.

wls:/weblogic/serverConfig> setBPELConnection(appName='webcenter',
name='WebCenter Worklist',url='http://myhost.com:6666', policy='oracle/wss10_saml_token_with_message_protection_client_policy')

The following example updates the BPEL server URL for a connection named
WebCenter Worklist.

wls:/weblogic/serverConfig> setBPELConnection(appName='webcenter',
name='WebCenter Worklist',url='http://myhost.com:8001')

5.13.3 listBPELConnections
Module: Oracle WebCenter
Use with WLST: Online
5.13.3.1 Description

Without any arguments, this command lists all the BPEL connections that are configured for a specific WebCenter application. All BPEL connections are listed, even connections not currently used.

5.13.3.2 Syntax

listBPELConnections(appName, [verbose], [name], [server], [applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application for which you want to list BPEL server connections.</td>
</tr>
<tr>
<td>verbose</td>
<td>Optional. Displays BPEL server connection details in verbose mode. Valid options are true and false. When set to true, listBPELConnections lists all of the BPEL server connections that are configured, along with their details. When set to false, listBPELConnections lists connection names only. This argument defaults to false.</td>
</tr>
<tr>
<td>name</td>
<td>Optional. Name of an existing BPEL server connection. You can use this argument to view details about a specific connection. To list all the connections, omit the name argument.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.13.3.3 Examples

The following example lists the names of all the BPEL server connections that are configured for a WebCenter application.

wls:/weblogic/serverConfig> listBPELConnections(appName='webcenter')

------------------
WebCenter Worklist
------------------
------------------
Human Resources Worklist
------------------

The following example lists the names and details of all of the BPEL server connections that are configured for a WebCenter application.

wls:/weblogic/serverConfig> listBPELConnections(appName='webcenter', verbose=true)

------------------
WebCenter Worklist
------------------
Connection Name: WebCenter Worklist
PolicyURI:oracle/wss10_saml_token_client_policy
URL:http://myhost.com:8001
------------------
Human Resources Worklist
------------------
Connection Name: Human Resources Worklist
PolicyURI:oracle/wss10_saml_token_client_policy
5.13.4 addWorklistConnection

Module: Oracle WebCenter
Use with WLST: Online

5.13.4.1 Description
Enable an existing BPEL server connection for Worklist services. The Worklist service supports multiple connections so that WebCenter users can monitor and manage assignments and notifications from a range of BPEL servers.

The name must specify an existing BPEL server connection.

5.13.4.2 Syntax
addWorklistConnection(appName, name, [verbose, server, applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>name</td>
<td>Name of an existing BPEL server connection.</td>
</tr>
<tr>
<td>verbose</td>
<td>Optional. Displays output indicating whether a matching BPEL server connection exists and provides connection details. True turns verbose mode on; false turns verbose mode off. This argument defaults to false.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.13.4.3 Examples
The following example enables the Human Resources Worklist connection for the Worklist service.

```
> addWorklistConnection(appName='webcenter', name='Human Resources Worklist', verbose=true)
Human Resources Worklist successfully added to WorkList
```

```
> addWorklistConnection(appName='webcenter', name='Human Resources Worklist', verbose=true)
Human Resources Worklist successfully added to WorkList
```

The following example also enables the Human Resources Worklist connection for the Worklist service.

```
> addWorklistConnection(appName='webcenter', name='Human Resources Worklist', verbose=true)
Human Resources Worklist successfully added to WorkList
```
Human Resources Worklist

Connection Name: Human Resources Worklist
PolicyURI: oracle/oracle/wss10_saml_token_client_policy
URL: http://myhost.com:8888

5.13.5 removeWorklistConnection
Module: Oracle WebCenter
Use with WLST: Online

5.13.5.1 Description
Disables a BPEL server connection that is currently used by the Worklist service. Connection details are retained but the Worklist service no longer uses the connection specified.

5.13.5.2 Syntax
removeWorklistConnection(appName, name, [server, applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>name</td>
<td>Name of an existing BPEL server connection.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.13.5.3 Example
The following example disables the BPEL server connection named WebCenter Worklist for the Worklist service.

wls:/weblogic/serverConfig> removeWorklistConnection(appName='webcenter', 
name='WebCenter Worklist')
WebCenter Worklist successfully removed from WorkList

5.13.6 listWorklistConnections
Module: Oracle WebCenter
Use with WLST: Online

5.13.6.1 Description
Without any arguments, this command lists all of the BPEL server connections that are configured for the Worklist service, in a named WebCenter application.

5.13.6.2 Syntax
listWorklistConnections(appName, [verbose], [name], [server], [applicationVersion])
### 5.13.6.3 Examples

The following example lists the names of all of the BPEL server connections that are configured for the Worklist service.

```
$ wls:/weblogic/serverConfig> listWorklistConnections(appName='webcenter')
------------------
WebCenter Worklist
------------------
```

The following example lists both the names and connection details of all of the BPEL server connections that are configured for the Worklist service.

```
$ wls:/weblogic/serverConfig> listWorklistConnections(appName='webcenter', verbose=true)
------------------
WebCenter Worklist
------------------
  Connection Name: WebCenter Worklist
  PolicyURI: oracle/wss10_saml_token_client_policy
  URL: http://myhost.com:8001
```

The following example lists connection details of a named BPEL server connection—`MyWorklist`. As the Worklist service is not currently configured to use `MyWorklist`, an appropriate message displays.

```
$ wls:/weblogic/serverConfig> listWorklistConnections(appName='webcenter', verbose=true, name='MyWorklist')
------------------
The following connection is not in the ADF Worklist:MyWorklist
```

### 5.14 WebCenter Spaces Workflows

Use the commands listed in Table 5–21 to manage the BPEL server connection used for WebCenter Spaces workflows (group space membership notifications, group space subscription requests, and so on).
Configuration changes made using these WebCenter WLST commands are only effective after your restart the Managed Server on which the WebCenter Spaces deployed. For details, see Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter.

**Table 5–21  WebCenter Spaces Workflow WLST Commands**

<table>
<thead>
<tr>
<th>Use This Command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>getSpacesWorkflowConnectionName</td>
<td>Return the name of the BPEL server connection that WebCenter Spaces is using for internal workflows.</td>
<td>Online</td>
</tr>
<tr>
<td>setSpacesWorkflowConnectionName</td>
<td>Specify the BPEL server connection used for Webcenter Spaces workflows.</td>
<td>Online</td>
</tr>
</tbody>
</table>

#### 5.14.1 getSpacesWorkflowConnectionName

Module: Oracle WebCenter

Use with WLST: Online

**5.14.1.1 Description**

Returns the name of the BPEL server connection that WebCenter Spaces is currently using for internal workflows (group space membership notifications, group space subscription requests, and so on).

**5.14.1.2 Syntax**

getSpacesWorkflowConnectionName(appName, [server, applicationVersion])

**Argument Definition**

- **appName**: Name of the WebCenter Spaces application—always webcenter.
- **server**: Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.
- **applicationVersion**: Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

**5.14.1.3 Example**

The following example names the BPEL server connection that WebCenter Spaces is currently using for internal workflow.

```
> wls:/weblogic/serverConfig> getSpacesWorkflowConnectionName(appName='webcenter')
WorkflowConfigConnectionName: WebCenter-Worklist
```

#### 5.14.2 setSpacesWorkflowConnectionName

Module: Oracle WebCenter

Use with WLST: Online

**5.14.2.1 Description**

Specifies the BPEL server connection that Webcenter Spaces uses for internal workflows. WebCenter Spaces uses a BPEL server included with the Oracle SOA Suite to host internal workflows, such as group space membership notifications, group...
space subscription requests, and so on. The connection name specified here must be a valid BPEL server connection.

5.14.2.2 Syntax

`setSpacesWorkflowConnectionName(appName, name, [server, applicationVersion])`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter Spaces application—always <code>webcenter</code>.</td>
</tr>
<tr>
<td>name</td>
<td>Name of an existing BPEL connection.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, <code>WLS_Spaces</code>. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.14.2.3 Example

The following example specifies that WebCenter Spaces uses the BPEL server connection named `WebCenter-Worklist` for its internal workflows.

```
wlst:/weblogic/serverConfig> setSpacesWorkflowConnectionName(appName='webcenter', name='WebCenter-Worklist')
```

5.15 Wikis and Blogs

Use the commands listed in Table 5–22 to manage wiki and blog server connections for a WebCenter application.

Configuration changes made using these WebCenter WLST commands are only effective after your restart the Managed Server on which the WebCenter application is deployed. For details, see Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter.

<table>
<thead>
<tr>
<th>Table 5–22 Wiki and Blog WLST Commands</th>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>createWikiserverConnection</td>
<td>Create a wiki and blog server connection for a named WebCenter application.</td>
<td></td>
<td>Online</td>
</tr>
<tr>
<td>setWikiserverConnection</td>
<td>Edit an existing wiki and blog server connection.</td>
<td></td>
<td>Online</td>
</tr>
<tr>
<td>listWikiserverConnections</td>
<td>List wiki and blog server connections for a named WebCenter application.</td>
<td></td>
<td>Online</td>
</tr>
<tr>
<td>listDefaultWikiserverConnection</td>
<td>List the default connection used by the Wiki and Blog service.</td>
<td></td>
<td>Online</td>
</tr>
</tbody>
</table>

5.15.1 createWikiserverConnection

Module: Oracle WebCenter

Use with WLST: Online
5.15.1.1 Descriptions
Creates a wiki and blog server connection for a named WebCenter application

5.15.1.2 Syntax
createWikiserverConnection(appName, name, url, passcode [default, server, applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>name</td>
<td>Connection name. The name must be unique (across all connection types) within the WebCenter application.</td>
</tr>
<tr>
<td>url</td>
<td>Base URL for the wiki and blog server. Enter an Oracle WebCenter Wiki and Blog Server URL. Use the format: &lt;protocol&gt;://&lt;host&gt;:&lt;port&gt;</td>
</tr>
<tr>
<td>passcode</td>
<td>Passcode that is required to call methods in Oracle WebCenter Wiki and Blog Web Services. The passcode is an arbitrary string that the administrator sets up in Oracle WebCenter Wiki and Blog Server after installation to prevent unauthorized access. Contact the administrator to obtain the server’s passcode.</td>
</tr>
<tr>
<td>default</td>
<td>Optional. Indicates whether this connection is the default connection for the Wiki and Blog service. Valid values are true and false. The default for this argument is false. To specify that the Wiki and Blog service uses this connection, set the value to true. While you can register multiple connections for a WebCenter application, only one connection is used for wiki and blog services—the default (or active) connection.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.15.1.3 Example
The following example creates a connection named MyWikiServer with the server URL and passcode values provided:

```bash
wls:/weblogic/serverConfig> createWikiserverConnection(appName='webcenter', name='MyWikiServer', url='http://myhost.com:8001', passcode='mywiki', default=true)
```

5.15.2 setWikiserverConnection
Module: Oracle WebCenter
Use with WLST: Online

5.15.2.1 Descriptions
Edits an existing wiki and blog server connection for a named WebCenter application.
5.15.2.2 Syntax

```
setWikiserverConnection(appName, name, [url, passcode, default, server, applicationVersion])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>name</td>
<td>Connection name. The name must be unique (across all connection types) within the WebCenter application.</td>
</tr>
<tr>
<td>url</td>
<td>Optional. Base URL for the wiki and blog server. Enter an Oracle WebCenter Wiki and Blog Server URL. Use the format: <code>&lt;protocol&gt;://&lt;host&gt;:&lt;port&gt;</code></td>
</tr>
<tr>
<td>passcode</td>
<td>Optional. Passcode that is required to call methods in Oracle WebCenter Wiki and Blog Web Services. The passcode is an arbitrary string that the administrator sets up in Oracle WebCenter Wiki and Blog Server after installation to prevent unauthorized access. Contact the administrator to obtain the server's passcode.</td>
</tr>
<tr>
<td>default</td>
<td>Optional. Indicates whether this connection is the default connection for the Wiki and Blog service. Valid values are <code>true</code> and <code>false</code>. To specify that the Wiki and Blog service uses this connection, set the value to <code>true</code>. While you can register multiple connections for a WebCenter application, only one connection is used for wiki and blog services—the default (or active) connection.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, <code>WLS_Spaces</code>. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.15.2.3 Example

The following example updates the wiki and blog server URL for a connection named `MyWikiServer`:

```
wlst:/weblogic/serverConfig> setWikiserverConnection(appName='webcenter', name='MyWikiServer', url='http://myhost.com:6666', passcode='mywiki', default=true)
```

5.15.3 listWikiserverConnections

Module: Oracle WebCenter

Use with WLST: Online

5.15.3.1 Description

Lists all of the wiki and blog server connections that are configured for a named WebCenter application.

5.15.3.2 Syntax

```
listWikiserverConnections(appName, [verbose, name, server, applicationVersion])
```
5.15.3.3 Example
The following example lists connection names and details for all of the wiki and blog server connections currently configured for WebCenter Spaces.

```
wlst:/weblogic/serverConfig>listWikiserverConnections(appName='webcenter', verbose=true)
```

The following example displays connection details for a wiki and blog server connection named MyWikiConnection.

```
wlst:/weblogic/serverConfig>listWikiserverConnections(appName='webcenter', verbose=true, name='MyWikiConnection')
```

5.15.4 listDefaultWikiserverConnection
Module: Oracle WebCenter
Use with WLST: Online

5.15.4.1 Description
Lists the connection that the Wiki and Blog service is using, in a named WebCenter application. While you can register multiple wiki and blog server connections for a WebCenter application, the Wiki and Blog service only uses one connection — the default (or active) connection.

5.15.4.2 Syntax
```
listDefaultWikiserverConnection(appName, verbose, [server, applicationVersion])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>verbose</td>
<td>Optional. Displays connection details for the Wiki and Blog service in verbose mode. Valid options are true and false. When set to true, listWikiserverConnections lists all of the wiki and blog server connections that are configured for a WebCenter application, along with their details. When set to false, only connection names are listed. This argument defaults to false. When set to false, do not specify the name argument.</td>
</tr>
<tr>
<td>name</td>
<td>Optional. Name of an existing wiki and blog connection. Use this argument to view connection details for a specific wiki and blog server.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

Argument Definition

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
</tbody>
</table>
The following example lists the name and details of the connection that the Wiki and Blog service is using, in an application named webcenter.

```
wls://weblogic/serverConfig> listDefaultWikiserverConnection(appName='webcenter', verbose=true)
```

## 5.16 WebCenter Identity Store

Use the commands listed in Table 5–23 to configure options for searching a WebCenter application’s identity store.

### Table 5–23 WebCenter Identity Store WLST Commands

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>setWebCenterIdStoreSearchConfig</td>
<td>Modify configuration options for searching a WebCenter application’s identity store.</td>
<td>Online</td>
</tr>
<tr>
<td>listWebCenterIdStoreSearchConfig</td>
<td>List current configuration options for searching a WebCenter application’s identity store.</td>
<td>Online</td>
</tr>
</tbody>
</table>

### 5.16.1 setWebCenterIdStoreSearchConfig

Module: Oracle WebCenter

Use with WLST: Online

#### 5.16.1.1 Description

Modifies configuration options for searching a WebCenter application’s identity store. Use these settings to optimize identity store searches (for users and roles) in a WebCenter application.

Identity store search parameters are stored in `adf-config.xml`. If a search parameter is not specified, it is not modified.

#### 5.16.1.2 Syntax

```
setWebCenterIdStoreSearchConfig(appName, [narrowSearchTimeout, broadSearchTimeout, maxSearchFilters, maxFetchRecords, server, applicationVersion])
```
5.16.1.3 Example
The following example increases both identity store search timeouts.

```wls:/weblogic/serverConfig> setWebCenterIdStoreSearchConfig(appName='webcenter', narrowSearchTimeout=60000, broadSearchTimeout=100000);
```

The following example limits the maximum number of records returned to 100.

```wls:/weblogic/serverConfig> setWebCenterIdStoreSearchConfig(appName='webcenter', maxFetchRecords=100);
```

### Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>narrowSearchTimeout</td>
<td>Optional. The maximum time allowed (in ms) for small, simple searches, such as fetching a single user from the identity store. The out-of-the-box default is 30000ms.</td>
</tr>
<tr>
<td>broadSearchTimeout</td>
<td>Optional. The maximum time allowed (in ms) to return large result sets, such as returning users and roles that match a name pattern. The out-of-the-box default is 60000.</td>
</tr>
<tr>
<td>maxSearchFilters</td>
<td>Optional. The number of search filters allowed for the WebCenter application's identity store. The maximum allowed, out-of-the-box, is 100.</td>
</tr>
<tr>
<td>maxFetchRecords</td>
<td>Optional. The maximum number of records to be returned from each search query. The out-of-the-box default is 100.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

#### 5.16.2 listWebCenterIdStoreSearchConfig
**Module:** Oracle WebCenter

**Use with WLST:** Online

#### 5.16.2.1 Description
Lists current configuration options for searching the WebCenter application's identity store.

Identity store search parameters are stored in `adf-config.xml`. 
5.16.2.2 Syntax

listWebCenterIdStoreSearchConfig(appName,[server, applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.16.2.3 Example

The following example displays identity store search configuration information for a WebCenter application named webcenter.

wls:/weblogic/serverConfig>listWebCenterIdStoreSearchConfig(appName='webcenter');

-----------------
User role search configuration parameters
-----------------
Narrow search timeout    : 30000
Broad search timeout     : 60000
Maximum search filters   : 100
Maximum records to fetch : 200

5.17 WebCenter Import and Export

Use the commands listed in Table 5–24 to export and import WebCenter Spaces and producer metadata associated with custom WebCenter applications.

<table>
<thead>
<tr>
<th>Table 5–24 Import and Export WLST Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use this command... To...</td>
</tr>
<tr>
<td>exportWebCenterApplication</td>
</tr>
<tr>
<td>importWebCenterApplication</td>
</tr>
<tr>
<td>exportGroupSpaces</td>
</tr>
<tr>
<td>exportGroupSpaceTemplates</td>
</tr>
<tr>
<td>importGroupSpaces</td>
</tr>
<tr>
<td>exportPortletClientMetadata</td>
</tr>
</tbody>
</table>
5.17.1 exportWebCenterApplication

Module: Oracle WebCenter

Use with WLST: Online

5.17.1.1 Description

(WebCenter Spaces only) Exports a WebCenter Spaces application to a WebCenter export archive (.EAR) using the filename provided.

5.17.1.2 Syntax

exportWebCenterApplication(appName, fileName, [exportCustomizations, exportSecurity, exportData, server, applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter Spaces application in which you want to perform this operation.</td>
</tr>
<tr>
<td>fileName</td>
<td>Name of the export archive EAR file to which you want the export to be written.</td>
</tr>
<tr>
<td>exportCustomizations</td>
<td>Optional. Valid values are true and false. When set to true, all customizations are exported. When set to false, customizations are not exported, that is, default task flows are exported without any customizations. This argument defaults to true.</td>
</tr>
<tr>
<td>exportSecurity</td>
<td>Optional. Valid values are true and false. When set to true, policy-store.xml contains application roles and permissions, as well as user details and their role assignments. When set to false, policy-store.xml contains application roles and permissions only. User details are not exported. This argument defaults to false.</td>
</tr>
<tr>
<td>exportData</td>
<td>Optional. Valid values are true and false. When set to true, data stored in the WebCenter Spaces database for lists, events, tags, and links is exported. When set to false, this data is not exported. This argument defaults to false.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.17.1.3 Examples

The following example exports a WebCenter Spaces application and all possible data to a file named myExport.ear.

wls:/weblogic/serverConfig> exportWebCenterApplication(appName='webcenter', fileName='myExport.ear', exportCustomizations=true, exportSecurity=true, exportData=true)
The following example exports a test application with the intention of importing the resultant EAR to an alternative system with a different user base. In this case, security policies (which might reference users or roles specific to the originating server) are not required. Additionally, data created during testing (such as lists, group space events, links, tags) is not required.

```wls:/weblogic/serverConfig> exportWebCenterApplication(appName='webcenter', fileName='export.ear')```

5.17.2 importWebCenterApplication

Module: Oracle WebCenter

Use with WLST: Online

5.17.2.1 Description

(WebCenter Spaces only) Imports a WebCenter Spaces application from a WebCenter export archive file to a server.

After importing WebCenter Spaces you will need to restart the managed server where the application is deployed.

5.17.2.2 Syntax

`importWebCenterApplication(appName, fileName, [server, applicationVersion])`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter Spaces application in which you want to perform this operation.</td>
</tr>
<tr>
<td>fileName</td>
<td>Name of the WebCenter export archive that you want to import.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.17.2.3 Example

The following example imports WebCenter Spaces from the export archive myExport.ear.

```wls:/weblogic/serverConfig> importWebCenterApplication(appName='webcenter', fileName='myExport.ear')```

5.17.3 exportGroupSpaces

Module: Oracle WebCenter

Use with WLST: Online

5.17.3.1 Description

(WebCenter Spaces only) Exports one or more group spaces to a WebCenter export archive (.EAR), using the filename specified.
5.17.3.2 Syntax

```plaintext
exportGroupSpaces(appName, fileName, names, [exportCustomizations, exportSecurity, exportData, server, applicationVersion])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>appName</code></td>
<td>Name of the WebCenter Spaces application in which you want to perform this operation—always <code>webcenter</code>.</td>
</tr>
<tr>
<td><code>fileName</code></td>
<td>Name of the local file to which the export will be written.</td>
</tr>
<tr>
<td><code>names</code></td>
<td>Names of the group spaces that you want to export. Separate multiple group space names with a comma.</td>
</tr>
<tr>
<td><code>exportCustomizations</code></td>
<td>Optional. Valid values are <code>true</code> and <code>false</code>. When set to <code>false</code>, customizations are not exported, that is, default task flows are exported without any customizations. This argument defaults to <code>true</code>.</td>
</tr>
<tr>
<td><code>exportSecurity</code></td>
<td>Optional. Valid values are <code>true</code> and <code>false</code>. When set to <code>true</code>, policy-store.xml contains group space roles and permissions, as well as member details and their role assignments. When set to <code>false</code>, policy-store.xml contains group space roles and permissions only. Member details are not exported. This argument defaults to <code>false</code>.</td>
</tr>
<tr>
<td><code>exportData</code></td>
<td>Optional. Valid values are <code>true</code> and <code>false</code>. When set to <code>true</code>, group space data stored in the WebCenter Spaces database for lists, events, tags, and links is exported. When set to <code>false</code>, this data is not exported. This argument defaults to <code>false</code>.</td>
</tr>
<tr>
<td><code>server</code></td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, <code>WLS_Spaces</code>. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td><code>applicationVersion</code></td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.17.3.3 Example

The following example exports two group spaces (myGroup1 and myGroup2) from a WebCenter Spaces application named `webcenter`.

```plaintext
wls:/weblogic/serverConfig> exportGroupSpaces(appName='webcenter', fileName='myExport.ear', names='myGroup1, myGroup2', exportCustomizations=true, exportSecurity=true, exportData=true)
```

5.17.4 `exportGroupSpaceTemplates`

Module: Oracle WebCenter

Use with WLST: Online

5.17.4.1 Description

(WebCenter Spaces only) Exports one or more group space templates to a WebCenter export archive (.EAR), using the filename specified.

5.17.4.2 Syntax

```plaintext
exportGroupSpaceTemplates(appName, fileName, names, [server, applicationVersion])
```
WebCenter Import and Export

5.17.4.3 Example
The following example exports two group space templates (myGroupTemplate1 and myGroupTemplate2) from a WebCenter Spaces application named webcenter.

```
wls:/weblogic/serverConfig> exportGroupSpaceTemplates(appName='webcenter', fileName='myExport.ear', names='myGroupTemplate1, myGroupTemplate2')
```

5.17.5 importGroupSpaces

Module: Oracle WebCenter
Use with WLST: Online

5.17.5.1 Description
(WebCenter Spaces only) Imports one or more group spaces or group space templates from a WebCenter export archive.

5.17.5.2 Syntax
```
importGroupSpaces(appName, fileName, [server, applicationVersion])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the WebCenter Spaces application in which you want to perform this operation—always webcenter.</td>
</tr>
<tr>
<td>fileName</td>
<td>Name of the WebCenter archive file that you want to import.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.17.5.3 Example
The following example imports group spaces or group space templates from a WebCenter archive named myExport.ear to a WebCenter Spaces application named webcenter.

```
wls:/weblogic/serverConfig> importGroupSpaces(appName='webcenter', fileName='myExport.ear')
```
5.17.6 exportPortletClientMetadata

Module: Oracle WebCenter
Use with WLST: Online

5.17.6.1 Description
Exports portlet client metadata and producer customizations and personalizations, for a custom WebCenter application. This command exports metadata for all the application's producers to a named export archive (.EAR file). You cannot opt to export metadata for specific producers.

Only use this command to migrate producer data associated with custom WebCenter applications. Do not use this command for WebCenter Spaces.

5.17.6.2 Syntax
exportPortletClientMetadata(appName, fileName, [exportPersonalizations, server, applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the custom WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>fileName</td>
<td>Name of the export archive (.EAR) to which you want the export to be written.</td>
</tr>
<tr>
<td>exportPersonalizations</td>
<td>Optional. Valid values are true and false. When set to true, personalizations for all producers are exported. When set to false, personalizations are not exported. This argument defaults to true.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

5.17.6.3 Example
The following example exports portlet client metadata and producer customizations and personalizations to an export archive named myExport.ear. Personalizations are not exported.

wls:/weblogic/serverConfig> exportPortletClientMetadata(appName='myApp', fileName='myExport.ear', exportPersonalizations=false)

5.17.7 importPortletClientMetadata

Module: Oracle WebCenter
Use with WLST: Online

5.17.7.1 Description
Imports portlet client metadata and producer customizations and personalizations from a named WebCenter export archive.

Producer personalizations are optional on export. Producer personalizations are imported if the export archive specified includes personalizations.
Only use this command to migrate producer data for a custom WebCenter application. Do not use this command for WebCenter Spaces.

### 5.17.7.2 Syntax

```bash
importPortletClientMetadata(appName, fileName, [server, applicationVersion])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appName</td>
<td>Name of the custom WebCenter application in which you want to perform this operation.</td>
</tr>
<tr>
<td>fileName</td>
<td>Name of the WebCenter export archive that you want to import.</td>
</tr>
<tr>
<td>server</td>
<td>Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.</td>
</tr>
</tbody>
</table>

### 5.17.7.3 Example

The following example imports portlet client metadata and producer customizations and personalizations from a WebCenter export archive named `myExport.ear`.  

```bash
wls:/weblogic/serverConfig> importPortletClientMetadata(appName='app1', fileName='myExport.ear')
```
Use the User Messaging Service commands, listed in Table 6–1, to download user messaging preferences from your backend database.

---

**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 *Oracle Fusion Middleware Administrator’s Guide.*

---

**Table 6–1  User Messaging Service for WLST Configuration**

<table>
<thead>
<tr>
<th>Command category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Section 6.1, “UMS WLST Command Group”</td>
<td>(Brief description of command category)</td>
</tr>
</tbody>
</table>

### 6.1  UMS WLST Command Group

The UMS WLST commands are listed under the command group “ums”.

#### 6.1.1  manageUserMessagingPrefs

**Command Category:** UMS  
**Use with WLST:** Offline

**6.1.1.1  Description**

`manageUserMessagingPrefs` 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.

**6.1.1.2  Syntax**

`manageUserMessagingPrefs {operation=, filename, url, username, password, [encoding], [guid], [merge] }`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>operation</td>
<td>specifies the upload or download operation to be performed.</td>
</tr>
</tbody>
</table>
6.1.1.3 Examples

To download the user messaging preferences of all users to the specified file.

```wls:offline> manageUserMessagingPrefs(operation='download', filename='download.xml', url='t3://localhost:8001', username='weblogic', password='<password>')```

To download the user messaging preferences of all users to the specified file using UTF-8 character encoding.

```wls:offline> manageUserMessagingPrefs(operation='download', filename='download.xml', url='t3://localhost:8001', 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> manageUserMessagingPrefs(operation='download', filename='download.xml', url='t3://localhost:8001', 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> manageUserMessagingPrefs(operation='download', filename='download.xml', url='t3://localhost:8001', 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> manageUserMessagingPrefs(operation='upload', filename='upload.xml', url='t3://localhost:8001', username='weblogic', password='<password>')```
To upload the user messaging preferences from the specified file to the backend database and overwrite existing preferences.

```wls:offline> manageUserMessagingPrefs(operation='upload', filename='upload.xml', url='t3://localhost:8001', username='weblogic', password='<password>', merge='overwrite')
```

### 6.1.2 deployUserMessagingDriver

Command Category: UMS

Use with WLST: Online

#### 6.1.2.1 Description

deployUserMessagingDriver is used to deploy additional instances of user messaging drivers.

Specify a base driver type (for example: email, xmpp, voicexml, and others) and a short name for the new driver deployment. The string `usermessagingdriver-` will be prepended to the specified application name. Any valid parameters for the `deploy` command can be specified, and will be passed through when the driver is deployed.

#### 6.1.2.2 Syntax

```
deployUserMessagingDriver(baseDriver, appName, [targets], [stageMode], [options])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseDriver</td>
<td>Specifies the base messaging driver type. Must be a known driver type, such as 'email', 'proxy', 'smpp', 'voicexml', or 'xmpp'.</td>
</tr>
<tr>
<td>appName</td>
<td>A short descriptive name for the new deployment. The specified value will be prepended with the string <code>usermessagingdriver-</code></td>
</tr>
<tr>
<td>targets</td>
<td>Optional. Additional arguments that are valid for the <code>deploy</code> command can be specified and will be passed through when the new driver is deployed.</td>
</tr>
<tr>
<td>stageMode</td>
<td></td>
</tr>
<tr>
<td>options</td>
<td></td>
</tr>
</tbody>
</table>

#### 6.1.2.3 Examples

To deploy a second instance of an email driver with name `myEmail`.

```wls:base_domain/servereConfig> deployUserMessagingDriver(baseDriver='email', appName='myEmail')
```

To deploy a second instance of an email driver, specifying deployment targets.

```wls:base_domain/servereConfig> deployUserMessagingDriver(baseDriver='email', appName='email2', targets='server1,server2')
```
7

## DMS Custom WLST Commands

Use the Dynamic Monitoring Service (DMS) commands to view 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 the *Oracle Fusion Middleware Administrator’s Guide*.

---

**Note:** To use these DMS custom WLST commands, you must invoke the WLST script from the Oracle Common home. See "Using Custom WLST Commands" in the *Oracle Fusion Middleware Administrator’s Guide*.

---

### 7.1 DMS Commands

Use the commands in Table 7–1 to view information about performance metrics.

<table>
<thead>
<tr>
<th>Table 7–1  DMS Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use this command...</strong></td>
</tr>
<tr>
<td>displayMetricTableNames</td>
</tr>
<tr>
<td>displayMetricTables</td>
</tr>
<tr>
<td>dumpMetrics</td>
</tr>
<tr>
<td>reloadMetricRules</td>
</tr>
</tbody>
</table>

#### 7.1.1 displayMetricTableNames

Use with WLST: Online

**7.1.1.1 Description**

Displays the names of the available DMS metric tables. The returned value is a string array containing metric table names.

**7.1.1.2 Syntax**

```bash
displayMetricTableNames([servers])
```
7.1.1.3 Examples
The following example displays metric table names from all WebLogic servers and system components:

displayMetricTableNames()
ADF
ADFc
ADFc_Metadata_Service
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 called soa_server1:

displayMetricTableNames(servers='soa_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=['soa_server1', 'bam-server1'])
ADF
ADFc
ADFc_Metadata_Service
ADFc_Region

Argument | Definition
--- | ---
servers | Optional. Specifies the servers from which to retrieve metrics. Valid values are a list of WebLogic server 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', ...]
servers=('servername1', 'servername2', ...)

If this argument is not specified, the command returns the list of metric table names from all WebLogic servers and system components.
7.1.2 displayMetricTables

Use with WLST: Online

7.1.2.1 Description
Displays the content of the DMS metric tables.

The returned value is an array of JMX javax.management.openmbean.CompositeData objects. Each array element has the following fields:

- The Table field is the metric table name.
- The Schema field is a javax.management.openmbean.TabularData object containing the metric table schema information.
- The Rows field is a javax.management.openmbean.TabularData object containing the metric table Rows.

The javax.management.openmbean.TabularData object for the metric table schema contains the following four fields:

- The Column field contains the name of the column.
- The Type field contains the type of the column value.
- The Unit field contains the unit of the column.
- The Description field contains the description of the column.

The javax.management.openmbean.TabularData object for the metric rows contains a field for every metric. It uses the metric name as the field name.

7.1.2.2 Syntax

```
displayMetricTables([metricTable_1], [metricTable_2], [...], [servers]
[variables])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>metricTable_n</td>
<td>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 name of the metric, without an argument name. You can specify zero or more metric table names, in a comma-separated list.</td>
</tr>
</tbody>
</table>
7.1.2.3 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 Server-0 and Server-2:

```java
displayMetricTables('JVM','weblogic.management.runtime.WebAppComponentRuntimeMBean',
    servers=['soa_server1','bam_server1'])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>servers</td>
<td>Optional. Specifies the servers from which to retrieve metrics. Valid values are a list of WebLogic server 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', ...] servers=('servername1', 'servername2', ...) If this argument is not specified, the command returns the list of metric tables from all WebLogic servers and system components.</td>
</tr>
<tr>
<td>variables</td>
<td>Optional. Defines the metric aggregation parameters. Valid values are a set of name-value pairs in a Jython map. 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.</td>
</tr>
</tbody>
</table>

The following example displays the aggregated metric tables with the specified metric aggregation parameters:

```java
displayMetricTables('j2ee_application:webservices_port_rollup',
    servers=['soa_server1','bam_server1'],
    variables={'host': 'hostname', 'servletName':'dms'})
```

7-4  Oracle Fusion Middleware WebLogic Scripting Tool Command Reference
Faults: 0
Requests: 0
Requests.averageTime: 0.0
Requests.totalTime: 0.0
ServerName: soa_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: soa_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.us.oracle.com
JVM: JVM
Name: threads
Parent: /JVM/MxBeans
Process: AdminServer:9001
ServerName: AdminServer
contention.value: enabled in JVM
daemon.value: 60 threads
deadlock.value: 0 threads
live.value: 61 threads
peak.value: 66 threads
started.value: 241 threads

Host: hostname.us.oracle.com
JVM: JVM
Name: threads
Parent: /JVM/MxBeans
Process: soa_server1:9001
7.1.3 dumpMetrics

Use with WLST: Online

7.1.3.1 Description
Displays available metrics in the internal format or in XML. The returned value is a text document.

7.1.3.2 Syntax

dumpMetrics([servers,] [format])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>servers</td>
<td>Optional. Specifies the servers from which to retrieve metrics. Valid values are a list of WebLogic server 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', ...] servers=('servername1', 'servername2', ...) If this argument is not specified, the command returns the list of metric tables from all WebLogic servers and system components.</td>
</tr>
<tr>
<td>format</td>
<td>Optional. Specifies the command output format. Valid values are 'raw' (the default) and 'xml'. For example: format='raw' format='xml' DMS raw format is a simple metric display format; it displays one metric per line.</td>
</tr>
</tbody>
</table>

7.1.3.3 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='bam_server1' cacheable='false'>
  <row cacheable='false'>
    <column name='serverName'><![CDATA[bam_server1]]></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>
  </row>...
The following example outputs metrics from Server-0 in the default raw format:

```python
dumpMetrics(servers='Server-0')
```

```java
/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://10.229.149.27:7001
state.value: RUNNABLE
waited.value: 0 msec
waitedCount.value: 0 times
```

The following example outputs metrics from Server-0 and Server-1 in XML format:

```python
dumpMetrics(servers=['soa_server1', 'bam_server1'], format='xml')
```

```xml
<table name='oracle_soainfra:high_latency_sync_composites' keys='ServerName soainfra_composite soainfra_composite_revision soainfra_domain' componentId='bam_server1' cacheable='false'>
</table>
<table name='weblogic_j2eeserver:ejb_transaction' keys='ServerName appName ejbModuleName name serverName' componentId='bam_server1' cacheable='false'>
<row cacheable='false'>
<column name='serverName'><![CDATA[bam_server1]]></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[bam_server1]]></column>
<column name='ejbTransactionCommit.throughput' type='DOUBLE'>0.0</column>
<column name='ejbModuleName'><![CDATA[sdpmessagingclient-ejb-parlayx.jar]]></column>
</row>
```

```python
dumpMetrics(servers=['soa_server1', 'bam_server1'], format='xml')
```

```xml
<table name='oracle_soainfra:high_latency_sync_composites' keys='ServerName soainfra_composite soainfra_composite_revision soainfra_domain' componentId='bam_server1' cacheable='false'>
</table>
<table name='weblogic_j2eeserver:ejb_transaction' keys='ServerName appName ejbModuleName name serverName' componentId='bam_server1' cacheable='false'>
<row cacheable='false'>
<column name='serverName'><![CDATA[bam_server1]]></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[bam_server1]]></column>
<column name='ejbTransactionCommit.throughput' type='DOUBLE'>0.0</column>
<column name='ejbModuleName'><![CDATA[sdpmessagingclient-ejb-parlayx.jar]]></column>
</row>
```
7.1.4 reloadMetricRules

Use with WLST: Online

7.1.4.1 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.

7.1.4.2 Syntax
reloadMetricRules([servers])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>servers</td>
<td>Optional. Specifies the servers that should reload their metric rules. Valid values are a list of WebLogic server names and system component names. To specify one server, use the following syntax: servers='servername' To specify multiple servers, use the following syntax: servers=['servername1', 'servername2', ...] servers=('servername1', 'servername2', ...) If this argument is not specified, the command reloads the metric rules for all WebLogic servers and system components.</td>
</tr>
</tbody>
</table>

7.1.4.3 Example
The following example reloads metric rules at the specified Managed Server:

    reloadMetricRules(servers='soa_server1')
Location changed to domainRuntime tree. This is a read-only tree with DomainMBean as the root.
For more help, use help(domainRuntime)

    loaded 'server-oracle_eps_server-11.0.xml'
    loaded 'server-weblogic_j2eeserver-11.0.xml'
    loaded 'server-oracle_bamweb-11.0.xml'
    loaded 'server-oracle_federation-11.0.xml'
    loaded 'server-portal-11.0.xml'
    loaded 'server-weblogic_j2ee_application_webcenter-11.0.xml'
    .
    .
    .
Use the logging commands to configure settings for log files and to view and search log files. Table 8–1 describes the different categories of logging commands.

For additional details about configuring and searching log files, see the chapter "Managing Log Files and Diagnostic Data" in the Oracle Fusion Middleware Administrator’s Guide.

---

**Note:** To use these logging custom WLST commands, you must invoke the WLST script from the Oracle Common home. See "Using Custom WLST Commands" in the Oracle Fusion Middleware Administrator’s Guide.

---

**Table 8–1 Logging Command Categories**

<table>
<thead>
<tr>
<th>Command category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log Configuration Commands</td>
<td>Configure settings for log files, such as the level of information written to the file or the maximum file size.</td>
</tr>
<tr>
<td>Search and Display Commands</td>
<td>View Oracle Fusion Middleware log files and search log files for particular messages.</td>
</tr>
</tbody>
</table>

---

### 8.1 Log Configuration Commands

Use the commands in Table 8–2 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 8–2 Logging Configuration Commands**

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
</tr>
</thead>
<tbody>
<tr>
<td>configureLogHandler</td>
<td>Configure an existing log handler, add a new handler, or remove existing handlers.</td>
</tr>
<tr>
<td>getLogLevel</td>
<td>Get the level for a given logger.</td>
</tr>
<tr>
<td>listLoggers</td>
<td>Get the list of loggers and the level of each logger.</td>
</tr>
<tr>
<td>listLogHandlers</td>
<td>List the configuration of one of more log handlers.</td>
</tr>
<tr>
<td>setLogLevel</td>
<td>Set the level for a given logger.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use with WLST...</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Online</td>
<td></td>
</tr>
<tr>
<td>Online</td>
<td></td>
</tr>
<tr>
<td>Online</td>
<td></td>
</tr>
<tr>
<td>Online</td>
<td></td>
</tr>
<tr>
<td>Online</td>
<td></td>
</tr>
</tbody>
</table>

---
8.1.1 configureLogHandler

Command Category: Log Configuration

Use with WLST: Online

8.1.1.1 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.

8.1.1.2 Syntax
configureLogHandler(options)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>options</td>
<td>Comma-separated list of options, specified as name-value pairs. Valid options include:</td>
</tr>
<tr>
<td></td>
<td>■ target—The name of a WebLogic server, 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.</td>
</tr>
<tr>
<td></td>
<td>■ name—The name of a log handler. This option is required.</td>
</tr>
<tr>
<td></td>
<td>■ maxFileSize—The value of the maxFileSize attribute 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.</td>
</tr>
<tr>
<td></td>
<td>■ maxLogSize—The value of the maxLogSize attribute 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).</td>
</tr>
<tr>
<td></td>
<td>■ rotationFrequency—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.</td>
</tr>
<tr>
<td></td>
<td>■ baseRotationTime—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.</td>
</tr>
<tr>
<td></td>
<td>■ retentionPeriod—The amount of time, in minutes, 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.</td>
</tr>
<tr>
<td></td>
<td>■ format—The format for the ODL handler. Valid values are one of the following strings: &quot;ODL-Text&quot; or &quot;ODL-XML&quot;. The default format is ODL-Text.</td>
</tr>
</tbody>
</table>
### 8.1.1.3 Examples

The following example specifies the maximum file size for the odl-handler:

```java
configureLogHandler(name="odl-handler", maxFileSize="5M")
```

The following example specifies the rotation frequency for the odl-handler:

```java
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 and maxLogSize:

```java
configureLogHandler(name="odl-handler", rotationFrequency="daily",
                      retentionPeriod="week", removeProperty=["maxFileSize","maxLogSize"])```

### 8.1.2 getLogLevel

**Command Category:** Log Configuration  
**Use with WLST:** Online

**8.1.2.1 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.

### 8.1.2.2 Syntax

getLogLevel(options)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>options</td>
<td>Comma-separated list of options, specified as name-value pairs. Valid options include:</td>
</tr>
<tr>
<td></td>
<td>■ target—The name of a WebLogic server, 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.</td>
</tr>
<tr>
<td></td>
<td>■ logger—A logger name. An empty string denotes the root logger. This option is required and has no default.</td>
</tr>
<tr>
<td></td>
<td>■ runtime—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).</td>
</tr>
</tbody>
</table>

### 8.1.2.3 Examples

The following example returns the level for the logger oracle:

getLogLevel(logger='oracle')

The following example returns the level for the logger oracle, specifying only config loggers:

getLogLevel(logger='oracle', runtime=0)

The following example returns the level for the logger oracle on the Oracle WebLogic Server server2:

getLogLevel(logger='oracle', target='server2')

### 8.1.3 listLoggers

Command Category: Log Configuration

Use with WLST: Online

#### 8.1.3.1 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.

#### 8.1.3.2 Syntax

listLoggers([[options]])
8.1.3.3 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")
```

8.1.4 listLogHandlers

Command Category: Log Configuration

Use with WLST: Online

8.1.4.1 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.

8.1.4.2 Syntax

```
listLogHandlers([options])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>options</td>
<td>An optional comma-separated list of options, specified as name-value pairs.</td>
</tr>
<tr>
<td></td>
<td>Valid options include:</td>
</tr>
<tr>
<td></td>
<td>- target—The name of a WebLogic server, or a string describing a system</td>
</tr>
<tr>
<td></td>
<td>component. For system components, refer to the component’s documentation</td>
</tr>
<tr>
<td></td>
<td>for details. The default value is the server to which WLST is connected.</td>
</tr>
<tr>
<td></td>
<td>- pattern—A regular expression pattern that is used to filter logger names.</td>
</tr>
<tr>
<td></td>
<td>The default value returns all logger names.</td>
</tr>
<tr>
<td></td>
<td>- runtime—A Jython boolean value (0 or 1) that determines if the operation</td>
</tr>
<tr>
<td></td>
<td>is to list runtime loggers or config loggers. The default value is 1.</td>
</tr>
</tbody>
</table>
8.1.4.3 Examples
The following example lists all log handlers:

```java
listLogHandlers()
```

The following example lists all log handlers named odl-handler:

```java
listLogHandlers(name="odl-handler")
```

The following example lists all log handlers for the WebLogic server server1:

```java
listLogHandlers(target="server1")
```

8.1.5 setLogLevel

Command Category: Log Configuration

Use with WLST: Online

8.1.5.1 Description
Sets the level of information written by a given Java logger to a log file.

8.1.5.2 Syntax

```java
setLogLevel(options)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>options</td>
<td>Comma-separated list of options, specified as name-value pairs. Valid options include:</td>
</tr>
<tr>
<td></td>
<td>■ target—The name of a WebLogic server, or a string describing a system component. For system components, refer to the component’s documentation for details.</td>
</tr>
<tr>
<td></td>
<td>The default value is the server to which WLST is connected.</td>
</tr>
<tr>
<td></td>
<td>■ logger—A logger name. An empty string denotes the root logger.</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>■ addLogger—A Jython boolean value (0 or 1) that determines if the logger should be created if it does not exist.</td>
</tr>
<tr>
<td></td>
<td>■ 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,</td>
</tr>
<tr>
<td></td>
<td>OR FINEST. Valid ODL levels include a message type followed by a colon and a message level. The valid ODL message types are: INCIDENT_</td>
</tr>
<tr>
<td></td>
<td>ERROR, ERROR, WARNING, NOTIFICATION, TRACE, and UNKNOWN. The message level is represented by an integer value that qualifies the message type.</td>
</tr>
<tr>
<td></td>
<td>Possible values are from 1 (highest severity) through 32 (lowest severity).</td>
</tr>
<tr>
<td></td>
<td>This option is required; there is no default value.</td>
</tr>
<tr>
<td></td>
<td>■ runtime—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</td>
</tr>
<tr>
<td></td>
<td>changing runtime loggers, this option is ignored.</td>
</tr>
<tr>
<td></td>
<td>■ persist—A Jython boolean value (0 or 1) that determines if the level should be saved to the configuration file. The default value is 1.</td>
</tr>
</tbody>
</table>
8.1.5.3 Examples
The following example sets the log level to NOTIFICATION:1 for the logger oracle.my.logger:

```python
gsetLogLevel(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:

```python
gsetLogLevel(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:

```python
gsetLogLevel(target="server1", logger="oracle.my.logger", level="WARNING", runtime=0)
```

8.2 Search and Display Commands
Use the commands in Table 8–3 to view Oracle Fusion Middleware log files and to search log files for particular messages.

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>displayLogs</td>
<td>List the logs for one or more components.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td>listLogs</td>
<td>Search and display the contents of log files.</td>
<td>Online or Offline</td>
</tr>
</tbody>
</table>

8.2.1 displayLogs
Command Category: Search and Display
Use with WLST: Online or Offline

8.2.1.1 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.

8.2.1.2 Syntax
`displayLogs([searchString,][options])`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>searchString</td>
<td>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.</td>
</tr>
</tbody>
</table>
**Search and Display Commands**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>options</strong></td>
<td>An optional comma-separated list of options, specified as name-value pairs. Valid options include:</td>
</tr>
<tr>
<td>■ <strong>target</strong>—The name of a WebLogic server, or a system component. For a system component, the syntax for the target is <code>opmn:instance-name/component-name</code> In connected mode, the default target is the WebLogic domain. In disconnected mode, there is no default; the target option is required.</td>
<td></td>
</tr>
<tr>
<td>■ <strong>oracleInstance</strong>—In disconnected mode, defines the path to the ORACLE_INSTANCE (or WebLogic domain home). In connected mode, this option is ignored.</td>
<td></td>
</tr>
<tr>
<td>■ <strong>log</strong>—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.</td>
<td></td>
</tr>
<tr>
<td>■ <strong>last</strong>—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).</td>
<td></td>
</tr>
<tr>
<td>■ <strong>tail</strong>—An integer value. Restrict the search to the last n messages from each log file and limits the number of messages displayed to n.</td>
<td></td>
</tr>
<tr>
<td>■ <strong>pattern</strong>—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.</td>
<td></td>
</tr>
<tr>
<td>■ <strong>ecid</strong>—A string or string sequence containing one or more Execution Context ID (ECID) values to be used as a filter for log messages.</td>
<td></td>
</tr>
<tr>
<td>■ <strong>component</strong>—A string or string sequence containing one or more component ID values to be used as a filter for log messages.</td>
<td></td>
</tr>
<tr>
<td>■ <strong>module</strong>—A string or string sequence containing one or more module ID values to be used as a filter for log messages.</td>
<td></td>
</tr>
<tr>
<td>■ <strong>type</strong>—A string or string sequence containing one or more message type values to be used as a filter for log messages.</td>
<td></td>
</tr>
<tr>
<td>■ <strong>app</strong>—A string or string sequence containing one or more application values to be used as a filter for log messages.</td>
<td></td>
</tr>
</tbody>
</table>
8.2.1.3 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='0000H19TwKUCs1T6uBi8UH18lkWX000002')

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="opmn:instance1/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")

8.2.2 listLogs

Command Category: Search and Display

Use with WLST: Online or Offline

8.2.2.1 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.

8.2.2.2 Syntax

listLogs([options]
**8.2.2.3 Examples**

The following example lists all of the log files for the WebLogic domain:

```java
listLogs()
```

The following example lists the log files for the WebLogic server server1:

```java
listLogs(target="server1")
```

The following example lists the log files for the Oracle HTTP Server ohs1:

```java
listLogs(target="opmn:instance1/ohs1")
```

The following example, used in disconnected mode, lists the log files for the WebLogic server server1:

```java
listLogs(oracleInstance="/middleware/user_projects/domains/base_domain", target="server1")
```

---

**Argument Definition**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>options</strong></td>
<td>An optional comma-separated list of options, specified as name-value pairs. Valid options include:</td>
</tr>
<tr>
<td>■ <strong>target</strong></td>
<td>The name of a WebLogic server, or an Oracle Fusion Middleware system component. For a system component, the syntax for the target is &quot;opmn:instance-name/component-name&quot;</td>
</tr>
<tr>
<td></td>
<td>In connected mode, the default target is the WebLogic domain. In disconnected mode, there is no default; the target option is required.</td>
</tr>
<tr>
<td>■ <strong>oracleInstance</strong></td>
<td>In disconnected mode, defines the path to the ORACLE_INSTANCE (or WebLogic domain home). In connected mode, this option is ignored.</td>
</tr>
<tr>
<td>■ <strong>unit</strong></td>
<td>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 &quot;ls -h&quot; option). The default value is H.</td>
</tr>
<tr>
<td>■ <strong>fullTime</strong></td>
<td>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.</td>
</tr>
</tbody>
</table>
Use the Oracle Metadata Services (MDS) commands in the categories listed in Table 9–1 to manage Oracle Metadata Services (MDS).

For additional details about creating and managing an MDS repository, see the chapter "Managing the Oracle Metadata Repository" in the Oracle Fusion Middleware Administrator’s Guide. For information about the roles needed to perform each operation, see "Understanding MDS Operations" in the Oracle Fusion Middleware Administrator’s Guide.

**Note:** To use these MDS custom WLST commands, you must invoke the WLST script from the Oracle Common home. See "Using Custom WLST Commands" in the Oracle Fusion Middleware Administrator’s Guide.

### Table 9–1  MDS Command Categories

<table>
<thead>
<tr>
<th>Command category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repository Management</td>
<td>Manage the MDS repository.</td>
</tr>
<tr>
<td>Commands</td>
<td></td>
</tr>
<tr>
<td>Application Metadata</td>
<td>Manage the application metadata in the MDS</td>
</tr>
<tr>
<td>Management Commands</td>
<td>repository.</td>
</tr>
<tr>
<td>Application Label Management</td>
<td>Manage the labels for the application.</td>
</tr>
<tr>
<td>Commands</td>
<td></td>
</tr>
<tr>
<td>Application Management</td>
<td>Manage the application deployment.</td>
</tr>
<tr>
<td>Deployment Commands</td>
<td></td>
</tr>
</tbody>
</table>

### 9.1 Repository Management Commands

Use the MDS commands listed in Table 9–2 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 9–2 Repository Management Commands**

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>createMetadataPartition</td>
<td>Create a metadata repository partition.</td>
<td>Online</td>
</tr>
<tr>
<td>deleteMetadataPartition</td>
<td>Delete a metadata repository partition.</td>
<td>Online</td>
</tr>
<tr>
<td>deregisterMetadataDBRepository</td>
<td>Deregister a database-based MDS repository.</td>
<td>Online</td>
</tr>
<tr>
<td>registerMetadataDBRepository</td>
<td>Register a database-based MDS repository.</td>
<td>Online</td>
</tr>
</tbody>
</table>

### 9.1.1 createMetadataPartition

Command Category: Repository Management

Use with WLST: Online

#### 9.1.1.1 Description

An application needs a metadata partition in the repository to manage its metadata. This command creates a partition with the given name in the specified repository. Each deployed application uses a logical partition in metadata repository. A metadata repository is used as a common repository for managing metadata of different applications. This logical partition also helps in maintaining the metadata lifecycle. Before deploying an application, you create a partition for it in MDS repository.

#### 9.1.1.2 Syntax

```plaintext
createMetadataPartition(repository, partition)
```

#### Argument Definition

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>repository</td>
<td>The name of the repository where the partition will be created.</td>
</tr>
<tr>
<td>partition</td>
<td>The name of the partition to create in the repository.</td>
</tr>
</tbody>
</table>

#### 9.1.1.3 Example

The following example creates the metadata partition "part1" in repository "mds-myrepos":

```plaintext
wls:/weblogic/serverConfig> createMetadataPartition(repository='mds-myrepos', partition='part1')

Executing operation: createMetadataPartition
Metadata partition created: part1

"part1"

wls:/weblogic/serverConfig>
```

### 9.1.2 deleteMetadataPartition

Command Category: Repository Management

Use with WLST: Online

#### 9.1.2.1 Description

Deletes a metadata partition in the specified repository. When you delete a repository partition, all of the metadata in that partition is lost.
9.1.2.2 Syntax

deleteMetadataPartition(repository, partition)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>repository</td>
<td>The name of the repository that contains the partition.</td>
</tr>
<tr>
<td>partition</td>
<td>The name of the partition to delete in the repository.</td>
</tr>
</tbody>
</table>

9.1.2.3 Example

The following example deletes the metadata partition "part1" from the repository "mds-myrepos":

```
wlst:/weblogic/serverConfig> deleteMetadataPartition(repository='mds-myrepos', partition='part1')
```

Executing operation: deleteMetadataPartition
Metadata partition deleted: part1
wlst:/weblogic/serverConfig>

---

9.1.3 deregisterMetadataDBRepository

Command Category: Repository Management
Use with WLST: Online

9.1.3.1 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.

9.1.3.2 Syntax

deregisterMetadataDBRepository(name)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the repository to deregister.</td>
</tr>
</tbody>
</table>

9.1.3.3 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.
wls:/weblogic/serverConfig>

---

9.1.4 registerMetadataDBRepository

Command Category: Repository Management
Use with WLST: Online

9.1.4.1 Description

A database metadata repository should be registered with WebLogic servers before the application can use it. This command registers a System JDBC data source with the domain for use as database-based metadata repository.
9.1.4.2 Syntax

registerMetadataDBRepository(name, dbVendor, host, port, dbName, user, password, [targetServers])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the repository to register.</td>
</tr>
<tr>
<td>dbVendor</td>
<td>The database vendor. The acceptable values are ORACLE or MSSQL.</td>
</tr>
<tr>
<td>host</td>
<td>The host name or the IP address of the database.</td>
</tr>
<tr>
<td>port</td>
<td>The port number used by the database.</td>
</tr>
<tr>
<td>dbName</td>
<td>The service name of the database. For example, orcl.hostname.com</td>
</tr>
<tr>
<td>user</td>
<td>The database user name.</td>
</tr>
<tr>
<td>password</td>
<td>The password for the database user.</td>
</tr>
<tr>
<td>targetServers</td>
<td>Optional. The WebLogic servers 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.</td>
</tr>
</tbody>
</table>

9.1.4.3 Example

The following example registers the metadata repository myrepos, to two servers, with the database parameters:

```
<weblogic> registerMetadataDBRepository('myrepos','ORACLE', 'test.oracle.com', '1521', 'mds', 'user1', 'x', 'server1, server2')
```

Executing operation: registerMetadataDBRepository.
Metadata DB repository 'mds-myrepos' was registered successfully.

```
<weblogic>
```

9.2 Application Metadata Management Commands

Use the commands in Table 9-3 to manage application metadata.

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>deleteMetadata</td>
<td>Deletes the metadata in the application repository.</td>
<td>Online</td>
</tr>
<tr>
<td>exportMetadata</td>
<td>Exports metadata for an application.</td>
<td>Online</td>
</tr>
<tr>
<td>importMetadata</td>
<td>Imports metadata for an application.</td>
<td>Online</td>
</tr>
<tr>
<td>purgeMetadata</td>
<td>Purge metadata.</td>
<td>Online</td>
</tr>
</tbody>
</table>

9.2.1 deleteMetadata

Command Category: Application Metadata

Use with WLST: Online

9.2.1.1 Description

Deletes the selected documents from the application repository. When this command is run against repositories that support versioning, that is a database-based repository,
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.

9.2.1.2 Syntax

deleteMetadata(application, server, docs, [restrictCustTo], [excludeAllCust], [excludeBaseDocs], [excludeExtendedMetadata], [cancelOnException], [applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>application</td>
<td>The name of the application for which the metadata is to be deleted.</td>
</tr>
<tr>
<td>server</td>
<td>The target server on which this application is deployed.</td>
</tr>
<tr>
<td>docs</td>
<td>A list of comma-separated, fully qualified document names or document name patterns, or both. The patterns can have the following wildcard characters: * and <strong>. The &quot;*&quot; represents all documents under the current namespace. The &quot;</strong>&quot; represents all documents under the current namespace and also recursively includes all documents in sub-namespaces. For example, &quot;/oracle/*&quot; will include all documents under &quot;/oracle/&quot; but not include documents under &quot;/oracle/mds/&quot;. As another example, &quot;/oracle/**&quot; will include all documents under &quot;/oracle/&quot; and also under &quot;/oracle/mds/&quot; and any other documents further in the namespace chain.</td>
</tr>
<tr>
<td>restrictCustTo</td>
<td>Optional. A list of comma-separated customization layer names used to restrict the delete operation so that it deletes only customization documents that match the specified customization layers. This argument will be ignored if the excludeAllCust argument is also specified.</td>
</tr>
<tr>
<td>excludeAllCust</td>
<td>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.</td>
</tr>
<tr>
<td>excludeBaseDocs</td>
<td>Optional. A Boolean value (true or false) that specifies whether or not to delete base documents. This argument defaults to false.</td>
</tr>
<tr>
<td>excludeExtendedMetadata</td>
<td>Optional. A Boolean value (true or false) that specifies whether or not to delete the Extended Metadata documents. This argument defaults to false.</td>
</tr>
<tr>
<td>cancelOnException</td>
<td>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.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. The application version, if multiple versions of the same application are deployed.</td>
</tr>
</tbody>
</table>

9.2.1.3 Examples

The following example deletes metadata files under the package "mypackage" from sampleApp deployed in the server "srg":
The following example deletes metadata files under the package "mypackage" from sampleApp deployed in the server "srg" and specifies to exclude extended metadata and all customizations:

```
<command>
  wls:/weblogic/serverConfig> deleteMetadata(application='mdsapp',
          server='srg', docs='/mypackage/*',
          cancelOnException='false',
          excludeExtendedMetadata='true',
          excludeAllCust='true')
</command>
```

```
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.
```

9.2.2 exportMetadata

Command Category: Application Metadata

Use with WLST: Online

**9.2.2.1 Description**

The application metadata can be transferred from one server location (for example, testing) to another server location (for example, production) by exporting and importing the metadata.

Use this command to export metadata.

**9.2.2.2 Syntax**

```
exportMetadata(application, server, toLocation, [docs], [restrictCustTo],
               [excludeAllCust], [excludeBaseDocs], [excludeExtendedMetadata], [fromLabel],
               [toLabel], [applicationVersion])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>application</td>
<td>The name of the application from which the metadata is to be exported.</td>
</tr>
<tr>
<td>server</td>
<td>The target server on which this application is deployed.</td>
</tr>
</tbody>
</table>
The following example exports all metadata files from the application "mdsapp" deployed in the server "srg".

```bash
wls:/weblogic/serverConfig> exportMetadata(application='mdsapp',
            server='srg', 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.

### Arguments Definition

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>toLocation</td>
<td>The target directory or archive file (.jar, JAR, .zip or .ZIP) to which documents selected from the source partition will be transferred. The directory or archive file must be a local or network directory or file where the application is physically deployed. If the location is a directory, but the directory does not exist, the directory will be created. If the location is an archive file, 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.</td>
</tr>
</tbody>
</table>
| 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 "*" represents all documents under the current namespace. 
 The "**" represents all documents under the current namespace and also recursively includes all documents in sub-namespaces. 
 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 thenamespace chain. |
| restrictCustTo     | Optional. 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. This argument will be 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 argument. |
| excludeBaseDocs    | Optional. A Boolean value (true or false) that specifies whether or not to export base documents. This argument defaults to false. |
| excludeExtendedMetadata | Optional. A Boolean value (true or false) that specifies whether or not to export the Extended Metadata documents. This argument defaults to 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. |

### 9.2.2.3 Examples

The following example exports all metadata files from the application "mdsapp" deployed in the server "srg".

```bash
wls:/weblogic/serverConfig> exportMetadata(application='mdsapp',
            server='srg', 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='srg', toLocation='/tmp/myrepos',
  restrictCustTo='user',
  excludeBaseDocs='true',
  fromLabel='label1',
  toLabel='label2',
  applicationVersion='11.1.1')
```

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/mdssys/cust/user/user1/write1.xml
/mypackage/mdssys/cust/user/user2/write2.xml
2 documents successfully transferred.

### 9.2.3 importMetadata

Command Category: Application Metadata

Use with WLST: Online

#### 9.2.3.1 Description

The application metadata can be transferred from one server location (for example, testing) to another server location (for example, production) by exporting and importing the metadata.

Use this command to import metadata.

#### 9.2.3.2 Syntax

```
importMetadata(application, server, fromLocation, [docs], [restrictCustTo],
  [excludeAllCust], [excludeBaseDocs], [excludeExtendedMetadata],
  [cancelOnException], [applicationVersion])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>application</td>
<td>The name of the application for which the metadata is to be imported.</td>
</tr>
<tr>
<td>server</td>
<td>The target server on which this application is deployed.</td>
</tr>
<tr>
<td>fromLocation</td>
<td>The source directory or archive file from which documents will be selected for transfer. The directory or archive file must be a local or network directory or file where the application is physically deployed. This argument can be used as a temporary file system location for transferring metadata from one server to another.</td>
</tr>
</tbody>
</table>
9.2.3.3 Example

The following example imports all metadata available in `/tmp/myrepos' to the application "mdsapp" deployed in the server "srg":

```
wls:/weblogic/serverConfig> importMetadata(application='mdsapp', server='srg', 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.

```
wls:/weblogic/serverConfig>
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>docs</td>
<td>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 <strong>. This argument defaults to &quot;/</strong>&quot;, which imports all of the documents in the repository. The &quot;<em>&quot; represents all documents under the current namespace. The &quot;**&quot; represents all documents under the current namespace and also recursively includes all documents in sub-namespaces. For example, &quot;/oracle/</em>&quot; will include all documents under &quot;/oracle/&quot; but not include documents under &quot;/oracle/mds/&quot;. &quot;/oracle/**&quot; will include all documents under &quot;/oracle/&quot; and also under &quot;/oracle/mds/&quot; and any other documents further in the namespace chain.</td>
</tr>
<tr>
<td>restrictCustTo</td>
<td>Optional. 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. This argument will be ignored if the excludeAllCust argument is also specified.</td>
</tr>
<tr>
<td>excludeAllCust</td>
<td>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.</td>
</tr>
<tr>
<td>excludeBaseDocs</td>
<td>Optional. A Boolean value (true or false) that specifies whether or not to import base documents. This argument defaults to false.</td>
</tr>
<tr>
<td>excludeExtendedMetadata</td>
<td>Optional. A Boolean value (true or false) that specifies whether or not to import the Extended Metadata documents. This argument defaults to false.</td>
</tr>
<tr>
<td>cancelOnException</td>
<td>Optional. A Boolean value (true or false) that specifies whether or not to abort the import operation when an exception is encountered.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. The application version, if multiple versions of the same application are deployed.</td>
</tr>
</tbody>
</table>

9.2.4 purgeMetadata

Command Category: Application Metadata

Use with WLST: Online
9.2.4.1 Description
Purges the unlabeled document's version from the application's repository. All documents will be purged if they are expired, based on Time-To-Live (the olderThan argument). Document versions that are attached to a label and those which are tip (latest version) are not purged. This command is applicable only for repositories that support versioning, that is, a database-based repository.

9.2.4.2 Syntax
```
purgeMetadata(application, server, olderThan, [applicationVersion])
```

### Argument Definition

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>application</td>
<td>The name of the application, used to identify the partition in the repository on which the purge operation will be run.</td>
</tr>
<tr>
<td>server</td>
<td>The target server on which this application is deployed.</td>
</tr>
<tr>
<td>olderThan</td>
<td>Document versions that are older than this value will be purged.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. The application version, if multiple versions of the same application are deployed.</td>
</tr>
</tbody>
</table>

9.2.4.3 Example
The following example purges the document version history for the application "mdsapp" deployed in the server "srg," if the version is older than 10 seconds:
```
wls:/weblogic/serverConfig> purgeMetadata('mdsapp', 'srg', 10)
Executing operation: purgeMetadata.
Metadata purged:Total number of versions: 10.
Number of versions purged: 0.
wls:/weblogic/serverConfig>
```

9.3 Application Label Management Commands
Use the commands in Table 9–4 to manage labels for applications.

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>createMetadataLabel</td>
<td>Creates a metadata label.</td>
<td>Online</td>
</tr>
<tr>
<td>deleteMetadataLabel</td>
<td>Deletes a metadata label from the repository partition</td>
<td>Online</td>
</tr>
<tr>
<td>listMetadataLabels</td>
<td>Lists metadata labels in the repository partition.</td>
<td>Online</td>
</tr>
<tr>
<td>promoteMetadataLabel</td>
<td>Promotes the metadata associated with a label to tip.</td>
<td>Online</td>
</tr>
</tbody>
</table>

9.3.1 createMetadataLabel
Command Category: Application Label Management
Use with WLST: Online

9.3.1.1 Description
Creates a new label for the documents in the application's repository partition. This command is applicable only for repositories that support versioning.
9.3.1.2 Syntax
createMetadataLabel(application, server, name, [applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>application</td>
<td>The name of the application for which a label will be created in the partition configured for this application.</td>
</tr>
<tr>
<td>server</td>
<td>The target server on which this application is deployed.</td>
</tr>
<tr>
<td>name</td>
<td>The name of the label to create in the repository partition.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. The application version, if multiple versions of the same application are deployed.</td>
</tr>
</tbody>
</table>

9.3.1.3 Example
The following example creates the label "label1" for the application "mdsapp" deployed in the server "srg":

```wsl
wls:/weblogic/serverConfig> createMetadataLabel('mdsapp','srg','label1')
Executing operation: createMetadataLabel.
Created metadata label "label1".
wls:/weblogic/serverConfig>
```

9.3.2 deleteMetadataLabel

Command Category: Application Label Management
Use with WLST: Online

9.3.2.1 Description
Deletes a label for the documents in the application's repository partition. This command is applicable only for repositories that support versioning.

9.3.2.2 Syntax
deleteMetadataLabel(application, server, name, [applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>application</td>
<td>The name of the application from whose associated partition the label is to be deleted.</td>
</tr>
<tr>
<td>server</td>
<td>The target server on which this application is deployed.</td>
</tr>
<tr>
<td>name</td>
<td>The name of the label to delete in the repository partition.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. The application version, if multiple versions of the same application are deployed.</td>
</tr>
</tbody>
</table>

9.3.2.3 Example
The following example deletes the metadata label "label1" from the application "mdsapp" deployed in the server "srg":

```wsl
wls:/weblogic/serverConfig> deleteMetadataLabel('mdsapp','srg','label1')
Executing operation: deleteMetadataLabel.
Deleted metadata label "label1".
wls:/weblogic/serverConfig>
```
9.3.3 listMetadataLabels

Command Category:
Use with WLST: Online

9.3.3.1 Description
Lists all of the metadata labels in the application’s repository partition. This command is applicable only for repositories that support versioning.

9.3.3.2 Syntax
listMetadataLabels(application, server, [applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>application</td>
<td>The name of the application for which all of the labels in the repository partition should be listed.</td>
</tr>
<tr>
<td>server</td>
<td>The target server on which this application is deployed.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. The application version, if multiple versions of the same application are deployed.</td>
</tr>
</tbody>
</table>

9.3.3.3 Example
The following example lists the metadata labels available for the application "mdsapp" deployed in the server "srg":

wls:/weblogic/serverConfig> listMetadataLabels('mdsapp', 'srg')
Executing operation: listMetadataLabels.
Database Repository partition contains the following labels:
label2
label3
wls:/weblogic/serverConfig>

9.3.4 promoteMetadataLabel

Command Category: Application Label Management
Use with WLST: Online

9.3.4.1 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.

9.3.4.2 Syntax
promoteMetadataLabel(application, server, name, [applicationVersion])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>application</td>
<td>The name of the application in whose associated repository the metadata is to be promoted to tip.</td>
</tr>
<tr>
<td>server</td>
<td>The target server on which this application is deployed.</td>
</tr>
<tr>
<td>name</td>
<td>The name of the label to promote in the repository partition.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. The application version, if multiple versions of the same application are deployed.</td>
</tr>
</tbody>
</table>
9.3.4.3 Example
The following example promotes the metadata label "label1" to tip in the application "mdsapp" deployed in the server "srg":

```
<weblogic>/serverConfig> promoteMetadataLabel('mdsapp', 'srg', 'label1')
```

Executing operation: promoteMetadataLabel.
Promoted metadata label "label1" to tip.

```
<weblogic>/serverConfig>
```

9.4 Application Management Deployment Commands

Use the commands in Table 9–5 to manage deployment.

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>getMDSArchiveConfig</td>
<td>Returns an MDSArchiveConfig object.</td>
<td>Offline</td>
</tr>
<tr>
<td>importMAR</td>
<td>Imports a MAR.</td>
<td>Online</td>
</tr>
</tbody>
</table>

9.4.1 getMDSArchiveConfig

Command Category: Application Management Deployment

Use with WLST: Offline

9.4.1.1 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 have 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 have 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.

### 9.4.1.2 Syntax

```java
archiveConfigObject = getMDSArchiveConfig(fromLocation)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>fromLocation</td>
<td>The name of the ear file, including its complete path.</td>
</tr>
</tbody>
</table>

The syntax for `setAppMetadataRepository` is:

```java
archiveConfigObject.setAppMetadataRepository([repository], [partition], [type], [jndi], [path])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>repository</td>
<td>Optional. The name of the application's repository.</td>
</tr>
<tr>
<td>partition</td>
<td>Optional. The name of the partition for the application's metadata.</td>
</tr>
<tr>
<td>type</td>
<td>Optional. The type of connection, file or database, to the repository. Valid values are 'File' or 'DB' (case insensitive).</td>
</tr>
<tr>
<td>jndi</td>
<td>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.</td>
</tr>
<tr>
<td>path</td>
<td>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.</td>
</tr>
</tbody>
</table>

The syntax for `setAppSharedMetadataRepository` is:

```java
archiveConfigObject.setAppSharedMetadataRepository(namespace, [repository], [partition], [type], [jndi], [path])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace</td>
<td>The namespace used for looking up the shared repository to set connection details.</td>
</tr>
<tr>
<td>repository</td>
<td>Optional. The name of the application's shared repository.</td>
</tr>
<tr>
<td>partition</td>
<td>Optional. The name of the partition for the application's shared metadata.</td>
</tr>
<tr>
<td>type</td>
<td>Optional. The type of connection, file or database, to the repository. Valid values are 'File' or 'DB' (case insensitive).</td>
</tr>
<tr>
<td>jndi</td>
<td>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.</td>
</tr>
<tr>
<td>path</td>
<td>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.</td>
</tr>
</tbody>
</table>

The syntax for `save` is:

```java
archiveConfigObject.save([toLocation])
```
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.

```java
archive = getMDSArchiveConfig(fromLocation='/tmp/testArchive.ear')
archive.setAppMetadataRepository(repository='AppRepos1', partition='partition1', type='DB', jndi='mds-jndi1')
archive.setAppSharedMetadataRepository(namespace='/a', repository='SharedRepos1', partition='partition2', type='File', path='/temp/dir')
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.

```java
archive = getMDSArchiveConfig(fromLocation='/tmp/testArchive.ear')
archive.setAppMetadataRepository(partition='MDS-partition2')
archive.setAppSharedMetadataRepository(namespace='/a', repository='SharedRepos2')
archive.save(toLocation='/tmp/targetArchive.ear')
```

## 9.4.2 importMAR

Command Category: Application Management Deployment

Use with WLST: Online

### 9.4.2.1 Description

Imports the metadata from the MAR packaged along 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.

### 9.4.2.2 Syntax

```java
importMAR(application, server, [applicationVersion])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>application</td>
<td>The name of the application for which the metadata is to be imported.</td>
</tr>
<tr>
<td>server</td>
<td>The target server on which this application is deployed.</td>
</tr>
<tr>
<td>applicationVersion</td>
<td>Optional. The application version, if multiple versions of the same application are deployed.</td>
</tr>
</tbody>
</table>

### 9.4.2.3 Example

The following example imports metadata from the MAR to the application "mdsapp":

```java
importMAR('mdsapp','srg')
```
Executing operation: importMAR.
"importMAR" operation completed. Summary of "importMAR" operation is:
 /appl/jobs.xml
 /appl/mo.xml
 2 documents successfully transferred.
wls:/weblogic/serverConfig>
This chapter describes WSLT commands for Oracle SOA Suite. These commands enable you to use WLST to configure SOA composite applications.

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 Oracle Fusion Middleware Administrator’s Guide.

This chapter includes the following sections:

- Section 10.1, "Overview of WSLT Command Categories"
- Section 10.2, "Deployment Commands"
- Section 10.3, "SOA Composite Application Management Commands"
- Section 10.4, "Configuration Plan Management Commands"
- Section 10.5, "Task Validation Commands"
- Section 10.6, "SOA Composite Application Compilation Commands"
- Section 10.7, "SOA Composite Application Packaging Commands"
- Section 10.8, "SOA Composite Application Test Commands"
- Section 10.9, "SOA Composite Application HTTP Client-Based Export and Import Commands"
- Section 10.10, "SOA Composite Application MBean-Based Export and Import Commands"

For additional details about deployment, configuration plans, and test suites, see Oracle Fusion Middleware Developer’s Guide for Oracle SOA Suite.

### 10.1 Overview of WSLT Command Categories

WLST commands are divided into the categories shown in Table 10–1.

<table>
<thead>
<tr>
<th>Table 10–1</th>
<th>Oracle SOA Suite Command Categories</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command category</td>
<td>Description</td>
</tr>
<tr>
<td>Section 10.2, &quot;Deployment Commands&quot;</td>
<td>Deploy and undeploy SOA composite applications.</td>
</tr>
</tbody>
</table>
Deployment Commands

10.2 Deployment Commands

Use the deployment commands, listed in Table 10–2, to deploy and undeploy SOA composite applications.

**Table 10–2  Deployment Commands for WLST Configuration**

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>sca_deployComposite</td>
<td>Deploy a SOA composite application.</td>
<td>Offline</td>
</tr>
<tr>
<td>sca_undeployComposite</td>
<td>Undeploy a SOA composite application.</td>
<td>Offline</td>
</tr>
</tbody>
</table>

10.2.1 sca_deployComposite

Command Category: Deployment Commands

Use with WLST: Offline

**10.2.1 Description**

Deploys a SOA composite application to the Oracle WebLogic Server. This command does not package the artifact files of the application for deployment. See Section 10.7, "SOA Composite Application Packaging Commands" for instructions on packaging a SOA composite application.

**10.2.1.2 Syntax**

`sca_deployComposite(serverURL, sarLocation, [overwrite], [user], [password], [forceDefault], configplan)`
### 10.2.1.3 Examples

The following example deploys the HelloWorld application.

```
wls:/mydomain/ServerConfig> sca_deployComposite("http://myhost10:7001", 
"/tmp/sca_HelloWorld_rev1.0.jar")
```

The following example deploys the HelloWorld application as the default version.

```
wls:/mydomain/ServerConfig> sca_deployComposite("http://myhost10:7001", 
"/tmp/sca_HelloWorld_rev1.0.jar", true)
```

The following example deploys the HelloWorld application with a required user name when basic authentication is configured. You are then prompted to provide the password for this user name.

```
wls:/mydomain/ServerConfig> sca_deployComposite("http://myhost10:7001", 
"/tmp/sca_HelloWorld_rev1.0.jar", user="weblogic")
Password:
```

The following example deploys the HelloWorld application and applies the configuration plan named deployplan.xml.

```
wls:/mydomain/ServerConfig> sca_deployComposite("http://myhost10:7001", 
"/tmp/sca_HelloWorld_rev1.0.jar", forceDefault=false, 
configplan="/tmp/deployplan.xml")
```
The following example deploys the HelloWorld ZIP file, which can include multiple SARs, MARs, or both.

```
wls:/mydomain/ServerConfig> sca_deployComposite("http://myhost:7001", 
"/tmp/HelloWorld.zip")
```

10.2.2 sca_undeployComposite

Command Category: Deployment Commands

Use with WLST: Offline

10.2.2.1 Description

Undeploys a currently deployed SOA composite application.

10.2.2.2 Syntax

```
sca_undeployComposite(serverURL, compositeName, revision, [user], [password])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>serverURL</td>
<td>URL of the server that hosts the SOA Infrastructure application (for example, <a href="http://myhost10:7001">http://myhost10:7001</a>).</td>
</tr>
<tr>
<td>compositeName</td>
<td>Name of the SOA composite application.</td>
</tr>
<tr>
<td>revision</td>
<td>Revision ID of the SOA composite application.</td>
</tr>
<tr>
<td>user</td>
<td>Optional. User name to access the composite deployer servlet when basic authentication is configured.</td>
</tr>
<tr>
<td>password</td>
<td>Optional. Password to access the composite deployer servlet when basic authentication is configured.</td>
</tr>
</tbody>
</table>

10.2.2.3 Examples

The following example undeploys the HelloWorld application.

```
wls:/mydomain/ServerConfig> sca_undeployComposite("http://myhost10:7001", 
"HelloWorld", "1.0")
```

The following example undeploys the HelloWorld application with a required user name when basic authentication is configured. You are then prompted to provide the password for this user name.

```
wls:/mydomain/ServerConfig> sca_undeployComposite("http://myhost10:7001", 
"HelloWorld", "1.0", user="weblogic")
Password:
```

10.3 SOA Composite Application Management Commands

Use the management commands, listed in Table 10-3, to start, stop, activate, retire, assign a default revision version, and list deployed SOA composite applications.
### Table 10–3 SOA Composite Application Management Commands for WLST Configuration

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>sca_startComposite</td>
<td>Start a previously stopped SOA composite application.</td>
<td>Offline</td>
</tr>
<tr>
<td>sca_stopComposite</td>
<td>Stop a SOA composite application.</td>
<td>Offline</td>
</tr>
<tr>
<td>sca_activateComposite</td>
<td>Activate a previously retired SOA composite application.</td>
<td>Offline</td>
</tr>
<tr>
<td>sca_retireComposite</td>
<td>Retire a SOA composite application.</td>
<td>Offline</td>
</tr>
<tr>
<td>sca_assignDefaultComposite</td>
<td>Assign the default revision version to a SOA composite application.</td>
<td>Offline</td>
</tr>
<tr>
<td>sca_listDeployedComposites</td>
<td>List the deployed SOA composite applications.</td>
<td>Offline</td>
</tr>
</tbody>
</table>

### 10.3.1 sca_startComposite

Command Category: Application Management Commands

Use with WLST: Offline

10.3.1.1 Description

Starts a previously stopped SOA composite application.

10.3.1.2 Syntax

`sca_startComposite(host, port, user, password, compositeName, revision, [label])`

10.3.1.3 Example

The following example starts revision 1.0 of the HelloWorld application.

```wls
wls:/mydomain/ServerConfig> sca_startComposite("myhost", "7001", "weblogic", "welcome1", "HelloWorld", "1.0")
```

### 10.3.2 sca_stopComposite

Command Category: Application Management Commands

Use with WLST: Offline

10.3.2.1 Description

Stops a SOA composite application.

10.3.2.2 Syntax

`sca_stopComposite(host, port, user, password, compositeName, [label])`

10.3.2.3 Example

The following example stops the HelloWorld application.

```wls
wls:/mydomain/ServerConfig> sca_stopComposite("myhost", "7001", "weblogic", "welcome1", "HelloWorld", "1.0")
```
10.3.2.1 Description
Stops a currently running SOA composite application.

10.3.2.2 Syntax
sca_stopComposite(host, port, user, password, compositeName, revision, [label])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>host</td>
<td>Hostname of the Oracle WebLogic Server (for example, myhost).</td>
</tr>
<tr>
<td>port</td>
<td>Port of the Oracle WebLogic Server (for example, 7001).</td>
</tr>
<tr>
<td>user</td>
<td>User name for connecting to the running server to get mBean information (for example, weblogic).</td>
</tr>
<tr>
<td>password</td>
<td>Password for the user name.</td>
</tr>
<tr>
<td>compositeName</td>
<td>Name of the SOA composite application.</td>
</tr>
<tr>
<td>revision</td>
<td>Revision of the SOA composite application.</td>
</tr>
<tr>
<td>label</td>
<td>Optional. Label of the SOA composite application. The label identifies the MDS artifacts associated with the application. If the label is not specified, the system finds the latest one.</td>
</tr>
</tbody>
</table>

10.3.2.3 Example
The following example stops revision 1.0 of the HelloWorld application.

```
> wls:/mydomain/ServerConfig> sca_stopComposite("myhost", "7001", "weblogic", "welcome1", "HelloWorld", "1.0")
```

10.3.3 sca_activateComposite
Command Category: Application Management Commands
Use with WLST: Offline

10.3.3.1 Description
Activates a retired SOA composite application and its instances. You can then create new instances.

10.3.3.2 Syntax
sca_activateComposite(host, port, user, password, compositeName, revision, [label])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>host</td>
<td>Hostname of the Oracle WebLogic Server (for example, myhost).</td>
</tr>
<tr>
<td>port</td>
<td>Port of the Oracle WebLogic Server (for example, 7001).</td>
</tr>
<tr>
<td>user</td>
<td>User name for connecting to the running server to get mBean information (for example, weblogic).</td>
</tr>
<tr>
<td>password</td>
<td>Password for the user name.</td>
</tr>
<tr>
<td>compositeName</td>
<td>Name of the SOA composite application.</td>
</tr>
<tr>
<td>revision</td>
<td>Revision of the SOA composite application.</td>
</tr>
</tbody>
</table>
10.3.3 Example
The following example activates revision 1.0 of the HelloWorld application.

```
wls:/mydomain/ServerConfig> sca_activateComposite("myhost", "7001", "weblogic", "welcome1", "HelloWorld", "1.0")
```

10.3.4 sca_retireComposite

Command Category: Application Management Commands
Use with WLST: Offline

10.3.4.1 Description
Stops and retires a SOA composite application and all its running instances. If the process life cycle is retired, you cannot create a new instance. Existing instances are allowed to complete normally.

10.3.4.2 Syntax
```
sca_retireComposite(host, port, user, password, compositeName, revision, [label])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>host</td>
<td>Hostname of the Oracle WebLogic Server (for example, myhost).</td>
</tr>
<tr>
<td>port</td>
<td>Port of the Oracle WebLogic Server (for example, 7001).</td>
</tr>
<tr>
<td>user</td>
<td>User name for connecting to the running server to get mBean information (for example, weblogic).</td>
</tr>
<tr>
<td>password</td>
<td>Password for the user name.</td>
</tr>
<tr>
<td>compositeName</td>
<td>Name of the SOA composite application.</td>
</tr>
<tr>
<td>revision</td>
<td>Revision of the SOA composite application.</td>
</tr>
<tr>
<td>label</td>
<td>Optional. Label of the SOA composite application. The label identifies the MDS artifacts associated with the application. If the label is not specified, the system finds the latest one.</td>
</tr>
</tbody>
</table>

10.3.4.3 Example
The following example retires revision 1.0 of the HelloWorld application.

```
wls:/mydomain/ServerConfig> sca_retireComposite("myhost", "7001", "weblogic", "welcome1", "HelloWorld", "1.0")
```

10.3.5 sca_assignDefaultComposite

Command Category: Application Management Commands
Use with WLST: Offline
10.3.5.1 Description
Sets a SOA composite application revision as the default version. This revision is
instantiated when a new request comes in.

10.3.5.2 Syntax
sca_assignDefaultComposite(host, port, user, password, compositeName, revision)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>host</td>
<td>Hostname of the Oracle WebLogic Server (for example, myhost).</td>
</tr>
<tr>
<td>port</td>
<td>Port of the Oracle WebLogic Server (for example, 7001).</td>
</tr>
<tr>
<td>user</td>
<td>User name for connecting to the running server to get mBean information (for example, weblogic).</td>
</tr>
<tr>
<td>password</td>
<td>Password for the user name.</td>
</tr>
<tr>
<td>compositeName</td>
<td>Name of the SOA composite application.</td>
</tr>
<tr>
<td>revision</td>
<td>Revision of the SOA composite application.</td>
</tr>
</tbody>
</table>

10.3.5.3 Example
The following example sets revision 1.0 of the HelloWorld application as the default
version.

wls:/mydomain/ServerConfig> sca_assignDefaultComposite("myhost", "7001",
"weblogic", "welcome1", "HelloWorld", "1.0")

10.3.6 sca_listDeployedComposites
Command Category: Application Management Commands
Use with WLST: Offline

10.3.6.1 Description
Lists all SOA composite applications deployed to the SOA platform.

10.3.6.2 Syntax
sca_listDeployedComposites(host, port, user, password)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>host</td>
<td>Hostname of the Oracle WebLogic Server (for example, myhost).</td>
</tr>
<tr>
<td>port</td>
<td>Port of the Oracle WebLogic Server (for example, 7001).</td>
</tr>
<tr>
<td>user</td>
<td>User name for connecting to the running server to get mBean information (for example, weblogic).</td>
</tr>
<tr>
<td>password</td>
<td>Password for the user name.</td>
</tr>
</tbody>
</table>

10.3.6.3 Example
The following example lists all the deployed SOA composite applications on the server
myhost.

wls:/mydomain/ServerConfig> sca_listDeployedComposites('myhost', '7001',
'weblogic', 'welcome1')
10.4 Configuration Plan Management Commands

Use the configuration plan management commands, listed in Table 10–4, to attach, extract, generate, and validate configuration plans for SOA composite applications.

### Table 10–4 Configuration Plan Management Commands for WLST Configuration

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>sca_attachPlan</td>
<td>Attach the configuration plan file to the SOA composite application JAR file.</td>
<td>Offline</td>
</tr>
<tr>
<td>sca_extractPlan</td>
<td>Extract a configuration plan packaged with the JAR file for editing.</td>
<td>Offline</td>
</tr>
<tr>
<td>sca_generatePlan</td>
<td>Generate a configuration plan for editing.</td>
<td>Offline</td>
</tr>
<tr>
<td>sca_validatePlan</td>
<td>Validate the configuration plan.</td>
<td>Offline</td>
</tr>
</tbody>
</table>

10.4.1 sca_attachPlan

Command Category: Configuration Plan Management Commands

Use with WLST: Offline

10.4.1.1 Description

Attaches the configuration plan file to the SOA composite application file. If a plan already exists in the file, it is overwritten with this new plan.

10.4.1.2 Syntax

`sca_attachPlan(sar, configPlan, [overwrite], [verbose])`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>sar</td>
<td>Absolute path of the SAR file.</td>
</tr>
<tr>
<td>configPlan</td>
<td>Absolute path of the configuration plan file.</td>
</tr>
<tr>
<td>overwrite</td>
<td>Optional. Indicates whether to overwrite an existing configuration plan in the SAR file.</td>
</tr>
<tr>
<td></td>
<td><img src="false.png" alt="false (default): Does not overwrite the plan." /></td>
</tr>
<tr>
<td></td>
<td><img src="true.png" alt="true: Overwrites the plan." /></td>
</tr>
<tr>
<td>verbose</td>
<td>Optional. Indicates whether to print more information about the configuration plan attachment.</td>
</tr>
<tr>
<td></td>
<td><img src="true.png" alt="true (default): Prints more information." /></td>
</tr>
<tr>
<td></td>
<td><img src="false.png" alt="false: Does not print more information." /></td>
</tr>
</tbody>
</table>

10.4.1.3 Examples

The following example attaches the `configplan.xml` configuration plan file to the HelloWorld application.

```
wlst:/mydomain/ServerConfig> sca_attachPlan("/tmp/sca_HelloWorld_rev1.0.jar", 
"/tmp/configplan.xml")
```

The following example overwrites the existing configuration plan with `configplan.xml` file in the HelloWorld application.

```
wlst:/mydomain/ServerConfig> sca_attachPlan("/tmp/sca_HelloWorld_rev1.0.jar", 
"/tmp/configplan.xml", true)
```
"/tmp/configplan.xml", overwrite=true)

10.4.2 sca_extractPlan
Command Category: Configuration Plan Management Commands
Use with WLST: Offline

10.4.2.1 Description
Extracts a configuration plan packaged with the SOA composite application file for editing. This is an optional step. If no plan exists, this is the same as creating a new file with sca_generatePlan.

10.4.2.2 Syntax
sca_extractPlan(sar, configPlan, [overwrite], [verbose])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>sar</td>
<td>Absolute path of a SAR file.</td>
</tr>
<tr>
<td>configPlan</td>
<td>Absolute path of a configuration plan file to which to be extracted.</td>
</tr>
<tr>
<td>overwrite</td>
<td>Optional. Indicates whether to overwrite an existing configuration plan file in the SAR file.</td>
</tr>
<tr>
<td>verbose</td>
<td>Optional. Indicates whether to print more information about configuration plan extraction.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>overwrite</td>
<td>■ false (default): Does not overwrite the plan.</td>
</tr>
<tr>
<td></td>
<td>■ true: Overwrites the plan.</td>
</tr>
<tr>
<td>verbose</td>
<td>■ true (default): Prints more information.</td>
</tr>
<tr>
<td></td>
<td>■ false: Does not print more information.</td>
</tr>
</tbody>
</table>

10.4.2.3 Example
The following example extracts the configplan.xml file for editing from the HelloWorld application.

wls:/mydomain/ServerConfig> sca_extractPlan("/tmp/sca_HelloWorld_rev1.0.jar", "/tmp/configplan.xml")

The following example extracts the configplan.xml file for editing from the HelloWorld application. This command also overwrites the existing plan.

wls:/mydomain/ServerConfig> sca_extractPlan("/tmp/sca_HelloWorld_rev1.0.jar", "/tmp/configplan.xml", overwrite=true)

10.4.3 sca_generatePlan
Command Category: Configuration Plan Management Commands
Use with WLST: Offline

10.4.3.1 Description
Generates a configuration plan for editing.

10.4.3.2 Syntax
sca_generatePlan(configPlan, sar, composite, [overwrite], [verbose])
10.4.3.3 Examples

The following example generates the myplan.xml configuration plan file for the HelloWorld application.

```bash
wls:/mydomain/ServerConfig> sca_generatePlan(’/tmp/myplan.xml’, sar=’/tmp/sca_HelloWorld_rev1.0.jar’)
```

The following example generates the myplan2.xml configuration plan file for the HelloWorld application. The myplan2.xml file overwrites the existing plan.

```bash
wls:/mydomain/ServerConfig> sca_generatePlan(’/tmp/myplan2.xml’, composite=’/tmp/HelloWorld_rev1.0/composite.xml’, overwrite=true)
```

10.4.4 sca_validatePlan

Command Category: Configuration Plan Management Commands

Use with WLST: Offline

10.4.4.1 Description

Validates the configuration plan. This command identifies all search and replacement changes to be made during deployment. Use this option for debugging only.

10.4.4.2 Syntax

```bash
sca_validatePlan(reportFile, configPlan, [sar], [composite], [overwrite], [verbose])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>reportFile</td>
<td>Absolute path of the report file to be generated. Validation results are</td>
</tr>
<tr>
<td></td>
<td>written to this file.</td>
</tr>
<tr>
<td>configPlan</td>
<td>Absolute path of the configuration plan file.</td>
</tr>
<tr>
<td>sar</td>
<td>Optional. The absolute path of the SAR file.</td>
</tr>
<tr>
<td>composite</td>
<td>Optional. The absolute path of the composite.xml file in the expanded</td>
</tr>
<tr>
<td></td>
<td>(unzipped) SAR directory.</td>
</tr>
</tbody>
</table>
| overwrite    | Optional. Indicates whether to overwrite an existing configuration plan file:
|              | ■ false (default): Does not overwrite the plan.                             |
|              | ■ true: Overwrites the plan.                                                |
| verbose      | Indicates whether to print more information about plan generation:          |
|              | ■ true (default): Prints more information.                                  |
|              | ■ false: Does not print more information.                                   |
10.4.4.3 Examples
The following example validates the configplan.xml configuration plan file for the HelloWorld application.

```
wlst:/mydomain/ServerConfig> sca_validatePlan("/tmp/myreport.xml", 
"/tmp/configplan.xml", sar="/tmp/sca_HelloWorld_rev1.0.jar")
```

The following example validates the configplan.xml configuration plan file for the HelloWorld application. The configplan.xml plan overwrites the existing plan.

```
wls:/mydomain/ServerConfig> sca_validatePlan("/tmp/myreport.xml", 
"/tmp/configplan.xml",composite="/tmp/HelloWorld_rev1.0/composite.xml", 
overwrite=true)
```

10.5 Task Validation Commands
Use the task validation command, listed in Table 10–5, to validate human workflow tasks.

Table 10–5  Task Validation Command for WLST Configuration

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>sca_validateTask</td>
<td>Validate a human workflow task.</td>
<td>Offline</td>
</tr>
</tbody>
</table>

10.5.1 sca_validateTask

Command Category: Task Validation Commands
Use with WLST: Offline

10.5.1.1 Description
Validates a human workflow task contained in the .task file that you created when designing a human task in the Human Task Editor.

10.5.1.2 Syntax
```
sca_validateTask(taskFile, outXml, [displayLevel])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>taskFile</td>
<td>Absolute path to the task definition file (.task).</td>
</tr>
<tr>
<td>outXml</td>
<td>Absolute path to an output XML file.</td>
</tr>
<tr>
<td>displayLevel</td>
<td>Optional. The level of information to display. The default value is 1.</td>
</tr>
</tbody>
</table>
10.5.1.3 Example
The following example validates the WFTaskDefinition.task file of the human task.

```
wlst:/mydomain/ServerConfig> sca_validateTask("/tmp/WFTaskDefinition.task", "/tmp/out.xml", displayLevel=2)
```

10.6 SOA Composite Application Compilation Commands
Use the compilation commands, listed in Table 10–6, to compile SOA composite applications.

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>sca_setProp</td>
<td>Set JVM system properties.</td>
<td>Offline</td>
</tr>
<tr>
<td>sca_compile</td>
<td>Compile a SOA composite application.</td>
<td>Offline</td>
</tr>
</tbody>
</table>

10.6.1 sca_setProp
Command Category: Application Compilation Commands
Use with WLST: Offline

10.6.1.1 Description
Sets JVM system properties. This command can also set secure socket layer (SSL) system properties before using sca_deployComposite and sca_undeployComposite over SSL.

10.6.1.2 Syntax
```
sca_setProp(propName, propValue)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>propName</td>
<td>Property name.</td>
</tr>
<tr>
<td>propValue</td>
<td>Property value.</td>
</tr>
</tbody>
</table>

10.6.1.3 Example
The following example sets the property name and property value.

```
wls:/mydomain/ServerConfig> sca_setProp("oracle.home", "/scratch/myusername/beahome/AS11gR1SOA")
```

10.6.2 sca_compile
Command Category: Application Compilation Commands
Use with WLST: Offline

10.6.2.1 Description
Compiles a SOA composite application.
10.6.2.2 Syntax

`sca_compile(composite, [outXml], [error], [appHome], [displayLevel], [oracleHome])`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>composite</td>
<td>Absolute path of a composite file in the expanded (unzipped) SAR directory.</td>
</tr>
<tr>
<td>outXml</td>
<td>Optional. Absolute path of an output XML file.</td>
</tr>
<tr>
<td>error</td>
<td>Optional. Absolute path of an error file.</td>
</tr>
<tr>
<td>appHome</td>
<td>Optional. Absolute path of the application home directory. This property is required if you have shared data.</td>
</tr>
<tr>
<td>displayLevel</td>
<td>Optional. The level of information to display. The default value is 1.</td>
</tr>
<tr>
<td>oracleHome</td>
<td>Optional. The oracle.home property.</td>
</tr>
</tbody>
</table>

10.6.2.3 Examples

The following example compiles the `FirstComposite` application.

```
wls:/mydomain/ServerConfig> sca_compile("/tmp/FirstComposite_rev1.0/composite.xml", displayLevel=2)
```

The following example compiles the `FirstComposite` application and captures details in the `myout.xml` file. The `error.out` file captures any errors.

```
wls:/mydomain/ServerConfig> sca_compile("/tmp/FirstComposite_rev1.0/composite.xml", outXml="/tmp/myout.xml", error="error.out")
```

The following example compiles the `FirstComposite` application. The `oracleHome` property is set to find the `ant-sca-compile.xml` script.

```
wls:/mydomain/ServerConfig> sca_compile("/tmp/FirstComposite_rev1.0/composite.xml", displayLevel=2, oracleHome="/scratch/myusername/beahome/AS11gR1SOA")
```

10.7 SOA Composite Application Packaging Commands

Use the packaging command, listed in Table 10–7, to package SOA composite applications into a composite SAR file.

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sca_package</code></td>
<td>Package the SOA composite application files into a composite SAR file.</td>
<td>Offline</td>
</tr>
</tbody>
</table>

10.7.1 sca_package

Command Category: Application Packaging Commands
Use with WLST: Offline

10.7.1 Description
Packages the SOA composite application files into a composite SAR file. This command performs the following operations:

- Calls `sca_compile` to compile the composite artifacts in `${compositeDir}`.
- Calls `javac` to compile any source code under `${compositeDir}/src`.
- Replaces the revision in `${compositeDir}/composite.xml`.
- Packages the artifacts to create `sca_${compositeName}_rev${revision}.jar` in `${compositeDir}/deploy`.

**Note:** The `sca_package` command requires `oracle.home` to find the `ant-sca-package.xml` script. This must be set once. You can use the `scac_setProp` command or `oracleHome` property to set this property.

10.7.2 Syntax
`sca_package(compositeDir, compositeName, revision, [appHome], [oracleHome])`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>compositeDir</td>
<td>Absolute path of a directory that contains composite artifacts.</td>
</tr>
<tr>
<td>compositeName</td>
<td>Name of the composite.</td>
</tr>
<tr>
<td>revision</td>
<td>Revision ID of the composite.</td>
</tr>
<tr>
<td>appHome</td>
<td>Optional. Absolute path of the application home directory. This property is required if you have shared data.</td>
</tr>
<tr>
<td>oracleHome</td>
<td>Optional. The <code>oracle.home</code> property.</td>
</tr>
</tbody>
</table>

10.7.3 Examples
The following example packages the `OrderBookingComposite` application. The `appHome` property is set because this application uses shared data.

```
wlst:/mydomain/ServerConfig> sca_package("/tmp/app_data/OrderBookingComposite", "OrderBookingComposite", "1.0", appHome="/tmp/app_data")
```

The following example packages the `HelloSOAComposite` application.

```
wlst:/mydomain/ServerConfig> sca_package("/tmp/HelloSOAApplication/HelloSOAComposite", "HelloSOAComposite", "1.0")
```

The following example packages the `HelloSOAComposite` application. The `oracleHome` property is set to find the `ant-sca-compile.xml` script.

```
wlst:/mydomain/ServerConfig> sca_package("/tmp/HelloSOAApplication/HelloSOAComposite", "HelloSOAComposite", "1.0", oracleHome="/scratch/myusername/beahome/AS11gR1SOA")
```

10.8 SOA Composite Application Test Commands
Use the SOA composite application test command, listed in Table 10–8, to test a SOA composite applications.
10.8.1 sca_test

Command Category: Application Test Commands

Use with WLST: Offline

10.8.1.1 Description
Tests deployed SOA composite applications prior to deployment in a production environment. You create suites of tests in Oracle JDeveloper. The sca_test command calls ant-sca-test.xml.

10.8.1.2 Syntax

\[
\text{sca}\_\text{test}('\text{compositeName}', '\text{revision}', '\text{testsuiteName}', '\text{jndiPropFile}', \[\text{oracleHome='oracleHome'}\], \[\text{javaHome='javaHome'}\])
\]

10.8.1.3 Examples

The following example runs the OrderBookingMainTestsuite test suite.

```
wlsl/mydomain/ServerConfig> sca_test('OrderBookingComposite', '1.0', 'OrderBookingMainTestsuite', '/tmp/tmp-jndi.properties', oracleHome='/scratch/<user>/beahome/AS11gR1SOA/', javaHome='/scratch/<user>/beahome/jdk160_05')
```

10.9 SOA Composite Application HTTP Client-Based Export and Import Commands

Use the SOA composite application commands, listed in Table 10–9, to export and import SOA composite applications based on the HTTP client. The SOA Infrastructure must be running to use these commands.

Table 10–9  SOA Composite Application Export and Import Commands for WLST Configuration

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>sca_exportComposite</td>
<td>Export a SOA composite application into a SAR file.</td>
<td>Offline</td>
</tr>
</tbody>
</table>
10.9.1 sca_exportComposite

Command Category: Application Export and Import Commands
Use with WLST: Offline

10.9.1.1 Description
Exports a SOA composite application into a SAR file.

10.9.1.2 Syntax
`sca_exportComposite(serverURL, updateType, sarFile, compositeName, revision, [user], [password])`

Argument | Definition
--- | ---
`serverURL` | URL of the server that hosts the SOA Infrastructure application (for example, `http://stabc:8001`).
`updateType` | Type of postdeployment changes to be exported:
- `all`: Includes all postdeployment changes.
- `property`: Includes only property postdeployment changes (binding component properties, composite properties such as audit level settings and payload validation status, and policy attachments).
- `runtime`: Includes only runtime (rules dictionary and domain value maps (DVMs)) and metadata postdeployment changes.
`sarFile` | Absolute path of a SAR file to generate (a `.jar` file that begins with `sca_`).
`compositeName` | Name of the composite to export.
`revision` | Revision of the composite to export.
`user` | Optional. The user name for accessing the server when basic configuration is configured. Use the following syntax for this argument:
- `user='username'`
`password` | Optional. The password for accessing the server when basic configuration is configured. Use the following syntax for this argument:
- `password='password'`
10.9.1.3 Examples
The following example exports the composite without including any postdeployment changes.

```
wls:/offline/mydomain/ServerConfig> sca_exportComposite('http://stabc:8001', 'none', '/tmp/sca_HelloWorld_rev1.0.jar', 'HelloWorld', '1.0')
```

The following example exports a composite with all postdeployment updates.

```
wls:/offline/mydomain/ServerConfig> sca_exportComposite('http://stabc:8001', 'all', '/tmp/sca_HelloWorld_rev1.0-all.jar', 'HelloWorld', '1.0')
```

The following example exports a composite with property postdeployment updates.

```
wls:/offline/mydomain/ServerConfig> sca_exportComposite('http://stabc:8001', 'property', '/tmp/sca_HelloWorld_rev1.0-prop.jar', 'HelloWorld', '1.0')
```

The following example exports a composite with runtime/metadata postdeployment updates.

```
wls:/offline/mydomain/ServerConfig> sca_exportComposite('http://stabc:8001', 'runtime', '/tmp/sca_HelloWorld_rev1.0-runtime.jar', 'HelloWorld', '1.0')
```

10.9.2 sca_exportUpdates
Command Category: Application Export and Import Commands
Use with WLST: Offline

10.9.2.1 Description
Exports postdeployment changes of a SOA composite application into a JAR file.

10.9.2.2 Syntax
```
sca_exportUpdates(serverURL, updateType, jarFile, compositeName, revision, [user], [password])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>serverURL</td>
<td>URL of the server that hosts the SOA Infrastructure application (for example, <a href="http://stabc:8001">http://stabc:8001</a>).</td>
</tr>
<tr>
<td>updateType</td>
<td>The type of postdeployment changes to be exported.</td>
</tr>
<tr>
<td></td>
<td>■ all: Includes all postdeployment changes.</td>
</tr>
<tr>
<td></td>
<td>■ property: Includes only property postdeployment changes (binding component properties, composite properties such as audit level settings and payload validation status, and policy attachments).</td>
</tr>
<tr>
<td></td>
<td>■ runtime: Includes only runtime (rules dictionary and domain value maps (DVMs)) and metadata postdeployment changes.</td>
</tr>
<tr>
<td>jarFile</td>
<td>Absolute path of a JAR file to generate. sca_exportUpdates() creates a regular .jar file that cannot be imported using regular deployment commands. It must be imported by using sca_importUpdates().</td>
</tr>
<tr>
<td>compositeName</td>
<td>Name of the composite to export.</td>
</tr>
<tr>
<td>revision</td>
<td>Revision of the composite to export.</td>
</tr>
</tbody>
</table>
SOA Composite Application HTTP Client-Based Export and Import Commands

10.9.2.3 Examples
The following example exports all postdeployment updates.

```java
wls:/offline/mydomain/ServerConfig> sca_exportUpdates('http://stabc:8001', 'all', '/tmp/all-HelloWorld_rev1.0.jar','HelloWorld', '1.0')
```

The following example exports property postdeployment updates.

```java
wls:/offline/mydomain/ServerConfig> sca_exportUpdates('http://stabc:8001', 'property','/tmp/prop-HelloWorld_rev1.0.jar', 'HelloWorld', '1.0')
```

The following example exports runtime/metadata postdeployment updates.

```java
wls:/offline/mydomain/ServerConfig> sca_exportUpdates('http://stabc:8001', 'runtime','/tmp/runtime-HelloWorld_rev1.0.jar', 'HelloWorld', '1.0')
```

10.9.3 sca_importUpdates

Command Category: Application Export and Import Commands
Use with WLST: Offline

10.9.3.1 Description
Imports postdeployment changes of a SOA composite application.

10.9.3.2 Syntax
```
sca_importUpdates(serverURL, jarFile, compositeName, revision, [user], [password])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>serverURL</td>
<td>URL of the server that hosts the SOA Infrastructure application (for example, <a href="http://stabc:8001">http://stabc:8001</a>).</td>
</tr>
<tr>
<td>jarFile</td>
<td>Absolute path of a JAR file that contains postdeployment changes.</td>
</tr>
<tr>
<td>compositeName</td>
<td>Name of the composite to which the postdeployment changes are imported.</td>
</tr>
<tr>
<td>revision</td>
<td>Revision of the composite to which the postdeployment changes are imported.</td>
</tr>
<tr>
<td>user</td>
<td>Optional. The user name for accessing the server when basic configuration is configured. Use the following syntax for this argument: user='username'</td>
</tr>
</tbody>
</table>
SOA Composite Application HTTP Client-Based Export and Import Commands

10.9.3.3 Examples
The following example imports postdeployment changes of a SOA composite application.

```
  wls:/offline/mydomain/ServerConfig> sca_importUpdates('http://stabc:8001', '/tmp/all-HelloWorld_rev1.0.jar', 'HelloWorld', '1.0')
```

10.9.4 sca_exportSharedData
Command Category: Application Export and Import Commands

Use with WLST: Offline

10.9.4.1 Description
Exports shared data of a given pattern into a JAR file.

10.9.4.2 Syntax
```
sca_exportSharedData(serverURL, jarFile, pattern, [user], [password])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>serverURL</td>
<td>URL of the server that hosts the SOA Infrastructure application (for example, <a href="http://stabc:8001">http://stabc:8001</a>).</td>
</tr>
<tr>
<td>jarFile</td>
<td>Absolute path of a JAR file to generate.</td>
</tr>
</tbody>
</table>
| pattern    | The file pattern supported by MDS transfer APIs. Use the semicolon delimiter (;) if more than one pattern is specified. Exclude the shared data namespace /apps in the pattern. For example:
|            | /Project1/**;/Project2/** |
|            | This example exports all documents under /apps/Project1 and /apps/Project2. |
| user       | Optional. The user name for accessing the server when basic configuration is configured. Use the following syntax for this argument:
|            | user='username' |
| password   | Optional. The password for accessing the server when basic configuration is configured. Use the following syntax for this argument:
|            | password='password' |

10.9.4.3 Examples
The following example exports shared data of a given pattern into a JAR file.

```
wls:/offline/mydomain/ServerConfig> sca_exportSharedData('http://stabc:8001', '/tmp/MySharedData.jar', '/Project1/**')
```
10.9.5 sca_removeSharedData

Command Category: Application Export and Import Commands

Use with WLST: Offline

10.9.5.1 Description

Removes a top-level shared data folder, even if there are composites deployed in the service engine.

10.9.5.2 Syntax

`sca_removeSharedData(serverURL, folderName, [user], [password])`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>serverURL</td>
<td>URL of the server that hosts the SOA Infrastructure application (for example, <a href="http://stabc:8001">http://stabc:8001</a>).</td>
</tr>
<tr>
<td>folderName</td>
<td>The name of a top-level shared data folder to be removed.</td>
</tr>
<tr>
<td>user</td>
<td>Optional. The user name for accessing the server when basic configuration is configured. Use the following syntax for this argument: user='username'</td>
</tr>
<tr>
<td>password</td>
<td>Optional. The password for accessing the server when basic configuration is configured. Use the following syntax for this argument: password='password'</td>
</tr>
</tbody>
</table>

10.9.5.3 Examples

The following example removes the top-level shared data `Project1` folder.

`sca_removeSharedData('http://stabc:8001', 'Project1')`

10.10 SOA Composite Application MBean-Based Export and Import Commands

Use the deployment commands, listed in Table 10–10, to export and import SOA composite applications on the server-based composite store MBean (CompositeStoreMXBean).

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>sca_exportCompositeMb</td>
<td>Export a SOA composite application into a SAR file.</td>
<td>Online</td>
</tr>
<tr>
<td>sca_exportUpdatesMb</td>
<td>Export postdeployment changes of a SOA composite application into a JAR file.</td>
<td>Online</td>
</tr>
<tr>
<td>sca_importUpdatesMb</td>
<td>Import postdeployment changes of a SOA composite application.</td>
<td>Online</td>
</tr>
<tr>
<td>sca_exportSharedDataMb</td>
<td>Export shared data of a given pattern into a JAR file.</td>
<td>Online</td>
</tr>
</tbody>
</table>
If you use this option, note that the file generated in the export commands and the file read in the import command must be on the host where the server is running (either an Oracle WebLogic Administration Server or a managed SOA server).

The composite store MBean is registered as both a server runtime MBean of the SOA server and as a domain runtime MBean of the Oracle WebLogic Administration Server, which allows the import and export to continue working while SOA servers are down. Only WLST commands are provided for using the composite store MBean; there are no ant commands.

You must run the `connect()` command to connect to either a SOA server or an Oracle WebLogic Administration Server.

```
wls:offline>connect('weblogic', 'password', 't3://stabc:8001')
```

If you use the domain runtime MBean while the SOA servers are down, you must run the `domainRuntime()` command.

```
wls:offline>connect('weblogic', 'password', 't3://stabc:7001')
wls:/soainfra/serverConfig>domainRuntime()
```

### 10.10.1 `sca_exportCompositeMb`

**Command Category:** Application Export and Import Commands  
**Use with WLST:** Online  

**10.10.1.1 Description**

Exports a SOA composite application into a SAR file.

**10.10.1.2 Syntax**

`sca_exportCompositeMb(updateType, sarFile, compositeName, revision)`

**Argument Definition**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>updateType</code></td>
<td>Type of postdeployment changes to be exported:</td>
</tr>
<tr>
<td></td>
<td>- all: All postdeployment changes are included.</td>
</tr>
<tr>
<td></td>
<td>- property: Property changes are included (binding component properties, composite properties such as audit level settings and payload validation status, and policy attachments).</td>
</tr>
<tr>
<td></td>
<td>- runtime: Postdeployment runtime changes are included (rules dictionary and domain value maps (DVMs)).</td>
</tr>
<tr>
<td><code>sarFile</code></td>
<td>Absolute path of a SAR file to generate.</td>
</tr>
<tr>
<td><code>compositeName</code></td>
<td>Name of the composite to export.</td>
</tr>
<tr>
<td><code>revision</code></td>
<td>Revision of the composite to export.</td>
</tr>
</tbody>
</table>

**10.10.1.3 Examples**

This example exports composite without including any postdeployment changes.

```
wls:/mydomain/ServerConfig> sca_exportCompositeMb('none', '/tmp/sca_HelloWorld_rev1.0.jar', 'HelloWorld', '1.0')
```

This example exports a composite with all postdeployment updates.

```
wls:/mydomain/ServerConfig> sca_exportCompositeMb('all', '/tmp/sca_HelloWorld_rev1.0-all.jar', 'HelloWorld', '1.0')
```
This example exports a composite with property postdeployment updates.

```
wls:/mydomain/ServerConfig> sca.exportCompositeMb('property', '/tmp/sca_HelloWorld_rev1.0-prop.jar', 'HelloWorld', '1.0')
```

This example exports a composite with runtime/metadata postdeployment updates.

```
wls:/mydomain/ServerConfig> sca.exportCompositeMb('runtime', '/tmp/sca_HelloWorld_rev1.0-runtime.jar', 'HelloWorld', '1.0')
```

### 10.10.2 sca.exportUpdatesMb

**Command Category:** Application Export and Import Commands

**Use with WLST:** Online

**10.10.2.1 Description**

Exports postdeployment changes of a SOA composite application into a JAR file.

**10.10.2.2 Syntax**

```
sca.exportUpdatesMb(updateType, jarFile, compositeName, revision)
```

**Argument Definition**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>updateType</td>
<td>Type of postdeployment changes to be exported: all, property, or runtime.</td>
</tr>
<tr>
<td>jarFile</td>
<td>Absolute path of a JAR file to generate.</td>
</tr>
<tr>
<td>compositeName</td>
<td>Name of the composite to export.</td>
</tr>
<tr>
<td>revision</td>
<td>Revision of the composite to export.</td>
</tr>
</tbody>
</table>

**10.10.2.3 Examples**

The following example exports all postdeployment updates.

```
wls:/mydomain/ServerConfig> sca.exportUpdatesMb('all', '/tmp/all-HelloWorld_rev1.0.jar', 'HelloWorld', '1.0')
```

The following example exports property postdeployment updates.

```
wls:/mydomain/ServerConfig> sca.exportUpdatesMb('property', '/tmp/prop-HelloWorld_rev1.0.jar', 'HelloWorld', '1.0')
```

The following example exports runtime/metadata postdeployment updates.

```
wls:/mydomain/ServerConfig> sca.exportUpdatesMb('runtime', '/tmp/runtime-HelloWorld_rev1.0.jar', 'HelloWorld', '1.0')
```

### 10.10.3 sca.importUpdatesMb

**Command Category:** Application Export and Import Commands

**Use with WLST:** Online

**10.10.3.1 Description**

Imports postdeployment changes of a SOA composite application.
10.10.3.2 Syntax
sca_importUpdatesMb(jarFile, compositeName, revision)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>jarFile</td>
<td>Absolute path of a JAR file that contains postdeployment changes.</td>
</tr>
<tr>
<td>compositeName</td>
<td>Name of the composite to which the postdeployment changes are imported.</td>
</tr>
<tr>
<td>revision</td>
<td>Revision of the composite to which the postdeployment changes are imported.</td>
</tr>
</tbody>
</table>

10.10.3.3 Examples
The following example imports postdeployment changes of a SOA composite application.

```
> sca_importUpdatesMb('/tmp/all-HelloWorld_rev1.0.jar', 'HelloWorld', '1.0')
```

10.10.4 sca_exportSharedDataMb

Command Category: Application Export and Import Commands

Use with WLST: Online

10.10.4.1 Description
Exports shared data of a given pattern into a JAR file.

10.10.4.2 Syntax
sca_exportSharedDataMb(jarFile, pattern)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>jarFile</td>
<td>Absolute path of a JAR file to generate.</td>
</tr>
<tr>
<td>pattern</td>
<td>The file pattern supported by MDS transfer APIs. Use the semicolon delimiter (;) if more than one pattern is specified. Exclude the shared data namespace /apps in the pattern. For example:/Project1/<strong>;/Project2/</strong></td>
</tr>
</tbody>
</table>

This example exports all documents under /apps/Project1 and /apps/Project2.

10.10.4.3 Examples
This example exports shared data of given pattern into a JAR file

```
> sca_exportSharedDataMb('/tmp/MySharedData.jar', '/Project1/***')
```
The following sections describe the WLST custom commands and variables in detail. Topics include:

- Section 11.1, "Overview of WLST Command Categories"
- Section 11.2, "Commands for ADF-based URL Connections"

**Note:** To use these ADF custom WLST commands, you must invoke the WLST script from the Oracle Common home. See "Using Custom WLST Commands" in the Oracle Fusion Middleware Administrator’s Guide.

### 11.1 Overview of WLST Command Categories

WLST commands are divided into the following categories.

**Table 11–1   WLST Command Categories**

<table>
<thead>
<tr>
<th>Command Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Section 11.2, &quot;Commands for ADF-based URL Connections&quot;</td>
<td>Navigate the hierarchy of configuration or runtime beans and control the prompt display.</td>
</tr>
</tbody>
</table>

### 11.2 Commands for ADF-based URL Connections

Use the commands in Table 11–1 to managing URL-based connections.

**Table 11–2   Browse Commands for WLST Configuration**

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>adf_createFileUrlConnection</td>
<td>Create a new ADF File connection.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td>adf_createHttpUrlConnection</td>
<td>Create a new ADF URL connection.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td>adf_setURLConnectionAttributes</td>
<td>Set or edit the attributes of a newly created or existing ADF connection.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td>adf_setURLConnectionAttributes</td>
<td>Create a new connection.</td>
<td>Online or Offline</td>
</tr>
</tbody>
</table>
11.2.1 adf_createFileUrlConnection

Use with WLST: Online or Offline

11.2.1.1 Description

Creates a new connection based on the oracle.adf.model.connection.url.FileURLConnection connection class.

11.2.1.2 Syntax

adf_createFileURLConnection(appName, name, URL)

11.2.1.3 Example

adf_createFileURLConnection('myapp', 'tempDir', '/scratch/tmp')

11.2.2 adf_createHttpUrlConnection

Use with WLST: Online or Offline

11.2.2.1 Description

Use this command to create a new connection based on the oracle.adf.model.connection.url.HttpURLConnection connection type class.

11.2.2.2 Syntax

adf.createHttpURLConnection (appName, name, [URL], [authenticationType], [realm], [user], [password])

11.2.2.3 Example

adf_createHttpURLConnection('myapp', 'cnn', 'http://www.cnn.com')
11.2.3 adf_setURLConnectionAttributes
Use with WLST: Online or Offline

11.2.3.1 Description
Use this command to set or edit the attributes of a newly created or existing ADF connection.

11.2.3.2 Syntax
adf_setURLConnectionAttributes(appname, connectionname, attributes)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appname</td>
<td>Application name for which the connection that will be created.</td>
</tr>
<tr>
<td>connectionname</td>
<td>The name of the new connection.</td>
</tr>
<tr>
<td>attributes</td>
<td>The array containing attributes to set in key/value pairs.</td>
</tr>
</tbody>
</table>

11.2.3.3 Example
adf_setURLConnectionAttributes('myapp','cnn','ChallengeAuthenticationType:digest', 'AuthenticationRealm:XMLRealm'

11.2.4 adf_listUrlConnection
Use with WLST: Online or Offline

11.2.4.1 Description
Use this command to create a new connection based on the oracle.adf.model.connection.url.FileURLConnection connection class.

11.2.4.2 Syntax
adf_listURLConnection(appname, name, URL)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appname</td>
<td>Application name for which the connection will be created.</td>
</tr>
<tr>
<td>name</td>
<td>The name of the new connection.</td>
</tr>
<tr>
<td>URL</td>
<td>The URL associated with this connection.</td>
</tr>
</tbody>
</table>

11.2.4.3 Example
adf_createFileURLConnection ('myapp','tempDir','/scratch/tmp')
Portal custom WLST commands are extensions to the WLST commands and are specific to Oracle Portal. Table 12–1 lists the Portal custom WLST command categories.

For additional information about administration and configuration of Portal, see the Oracle Portal Configuration Guide.

Table 12–1 Portal WLST Command Categories

<table>
<thead>
<tr>
<th>Command category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Section 12.1, &quot;Database Access Descriptor Commands&quot;</td>
<td>Create, edit, or delete a general DAD or Portal DAD.</td>
</tr>
<tr>
<td>Section 12.2, &quot;Configuration Commands&quot;</td>
<td>The Configuration commands:</td>
</tr>
<tr>
<td></td>
<td>■ List and update the WebCache configuration and Oracle Internet Directory data</td>
</tr>
<tr>
<td></td>
<td>■ Configure the Portal cache, Portal Page Engine, and Portal mid-tier</td>
</tr>
<tr>
<td></td>
<td>■ List Portal site configuration.</td>
</tr>
</tbody>
</table>

**12.1 Database Access Descriptor Commands**

A Database Access Descriptor (DAD) is a set of values that specify how an application connects to an Oracle database to fulfill an HTTP request. The information in the DAD includes the user name (which also specifies the schema and the privileges), password, connect string, and globalization support language of the database.

There are two types of DADs: general DAD and portal DAD. An Oracle Portal middle tier uses a Portal DAD to access the Oracle Metadata Repository. For information about general DADs, refer to the Oracle Fusion Middleware Administrator’s Guide for Oracle HTTP Server.

Use the Database Access Descriptor commands listed in Table 12–2 to create, edit, or delete a Portal DAD from the WLST command-line scripting interface. Based on your actions, the `portal_dads.conf` file is updated.
### Table 12–2  Database Access Descriptor Commands for Portal WLST Configuration

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>listDads</td>
<td>List the parameters used by the Database Access Descriptors for configuration.</td>
<td>Online</td>
</tr>
<tr>
<td>createPortalDad</td>
<td>Create a Portal Database Access Descriptor.</td>
<td>Online</td>
</tr>
<tr>
<td>updatePortalDad</td>
<td>Update the attributes of a Portal Database Access Descriptor.</td>
<td>Online</td>
</tr>
<tr>
<td>deletePortalDad</td>
<td>Delete a Portal Database Access Descriptor.</td>
<td>Online</td>
</tr>
</tbody>
</table>

#### 12.1.1 listDads

Command Category: Database Access Descriptor Commands

Use with WLST: Online

**12.1.1.1 Description**

Lists the parameters specified in all the Database Access Descriptors (both general DADs and Portal DADs).

**12.1.1.2 Syntax**

```plaintext
listDads()
```

**12.1.1.3 Example**

The following example lists the various DADs in the domain.

```plaintext
listDads()
------------
/pls/portal1
Schema: h1user
Connect String: foo.oracle.com:1521:orcl
NLS Language: "AMERICAN_AMERICA.AL32UTF8"
```

#### 12.1.2 createPortalDad

Command Category: Database Access Descriptor Commands

Use with WLST: Online

**12.1.2.1 Description**

Creates a Portal Database Access Descriptor.

**12.1.2.2 Syntax**

```plaintext
createPortalDad (name, schema, password, [connect_string], nls_language)
```

**Argument | Definition**
---|---
name | Name of the Database Access Descriptor.
schema | The Portal database account user name.
password | The Portal database account password.
### 12.1.2.3 Example

The following example creates the portal1 Portal DAD based on the specified arguments.

```python
createPortalDad(name='portal1',schema='schema',password='welcome1',connect_string='foo.oracle.com:1521:orcl',nls_language='AMERICAN_AMERICA.AL32UTF8')
```

### 12.1.3 updatePortalDad

**Command Category:** Database Access Descriptor Commands

**Use with WLST:** Online

#### 12.1.3.1 Description

Updates the attributes of the Portal Database Access Descriptor.

#### 12.1.3.2 Syntax

```python
updatePortalDad (name, [schema], [password], [connect_string], [nls_language])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the Database Access Descriptor. This name cannot be changed during update.</td>
</tr>
<tr>
<td>schema</td>
<td>Optional. The Portal database account user name.</td>
</tr>
<tr>
<td>password</td>
<td>Optional. The Portal database account password.</td>
</tr>
<tr>
<td>connect_string</td>
<td>Optional. The connection string used to connect to a remote database. Connect string may be host name: port number: connect string. The connect string format may be ServiceNameFormat (host:port:database_service_name), SIDFormat (host:port:database_sid), or TNSFormat (TNS alias or the whole TNS entry).</td>
</tr>
<tr>
<td>nls_language</td>
<td>Optional. The globalization support language of the Portal database that is represented by this DAD. This setting overrides the NLS_LANG environment variable for a database session and defines some important globalization support properties of the response, including the response character set. Make sure that this language setting matches the NLS_LANG of the back-end database.</td>
</tr>
</tbody>
</table>
12.1.3.3 Example
The following example updates the portal1 Portal DAD based on the specified arguments.

```java
updatePortalDad(name='portal1',schema='user1',password='welcome2',connect_string='foo.oracle.com:1521:orcl',nls_language='AMERICAN_AMERICA.AL32UTF8')
```

12.1.4 deletePortalDad

**Command Category:** Database Access Descriptor Commands

**Use with WLST:** Online

12.1.4.1 Description
Deletes a Portal Database Access Descriptor.

12.1.4.2 Syntax
```
deletePortalDad(name)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the Portal Database Access Descriptor.</td>
</tr>
</tbody>
</table>

12.1.4.3 Example
The following example deletes the portal1 Portal DAD entry from the portal_dads.conf file.
```
deletePortalDad(name='portal1')
```

12.2 Configuration Commands

Use the Configuration commands in Table 12–3 to view and configure Portal cache, WebCache, Oracle Internet Directory data and so on.

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>configurePortalCache</td>
<td>Update the attributes of the Portal cache.</td>
<td>Online</td>
</tr>
<tr>
<td>configurePortalPageEngine</td>
<td>Update the attributes of the Portal mid-tier.</td>
<td>Online</td>
</tr>
<tr>
<td>listPortalWebcacheConfigAttributes</td>
<td>List the attributes of WebCache configuration.</td>
<td>Online</td>
</tr>
<tr>
<td>listPortalSiteConfigAttributes</td>
<td>List the attributes of Portal site configuration.</td>
<td>Online</td>
</tr>
<tr>
<td>listPortalOIDConfigAttributes</td>
<td>List the attributes of Oracle Internet Directory configuration.</td>
<td>Online</td>
</tr>
<tr>
<td>setPortalWebcacheConfig</td>
<td>Update the attributes of the WebCache configuration.</td>
<td>Online</td>
</tr>
<tr>
<td>setPortalOIDConfig</td>
<td>Update the attributes of the Oracle Internet Directory configuration.</td>
<td>Online</td>
</tr>
<tr>
<td>setPortalMidtierConfig</td>
<td>Update the attributes of the Portal mid-tier configuration.</td>
<td>Online</td>
</tr>
</tbody>
</table>
12.2.1 configurePortalCache

Command Category: Configuration Commands

Use with WLST: Online

12.2.1.1 Description

Portal cache is a file system-based cache for Oracle Portal pages and portlets. Portal cache supports validation-based caching and expiry-based caching. Portal cache consists of both Portal content cache and session cache.

This command updates the attributes of the Portal cache. These configuration details are maintained in the `<Middleware Home>/user_projects/domains/<DOMAIN_HOME>/servers/WLS_PORTAL/stage/portal/portal/configuration/portal_cache.conf` file.

12.2.1.2 Syntax

`configurePortalCache(enable, directory, total_size, max_size, cleanup_time, max_age)`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>enable</td>
<td>Optional. Enables (On) or disables (Off) portal content and session caching.</td>
</tr>
<tr>
<td>directory</td>
<td>Optional. The directory where cached content is stored. Make sure that this directory exists and has read-write access.</td>
</tr>
<tr>
<td>total_size</td>
<td>Optional. The total amount of disk space (in megabytes) that the Portal cache may use. The maximum value allowed is 4 GB.</td>
</tr>
<tr>
<td>max_size</td>
<td>Optional. The maximum size (in bytes) for all cached files. The maximum value allowed is 4 GB. Any dynamically generated content that exceeds this limit is not cached.</td>
</tr>
<tr>
<td>cleanup_time</td>
<td>Optional. The time at which to start the cleanup of the cache storage. Use the [Sunday-Saturday, Everyday, Everymonth][hh:mm] format to define the exact day and time in which cleanup should occur.</td>
</tr>
<tr>
<td>max_age</td>
<td>Optional. Maximum age of a single cached document. This setting ensures the cache system does not contain any old content. Old cache files are removed to make space for new cache files. The default is 30 days.</td>
</tr>
</tbody>
</table>

12.2.1.3 Example

The following example configures the Portal cache.

`configurePortalCache(enable=true, directory='/scratch/user.Installs/Inst_1/cache/PortalComponent/portal', total_size=10101010, max_size=12300033, cleanup_time='Everyday 11:00', max_age=20)`

12.2.2 configurePortalPageEngine

Command Category: Configuration Commands

Use with WLST: Online
12.2.2.1 Description
The Oracle Fusion Middleware Portal architecture is designed around a three-tier architecture that allows any browser to connect to it. This flexible architecture allows each component (browser, Oracle HTTP Server listener, Oracle Database 11g, and Oracle Portal) to be upgraded individually as required.

A part of the Oracle Portal middle tier, the Parallel Page Engine (PPE) is a servlet that runs under Oracle Containers for J2EE and services page requests. The PPE reads page metadata, calls providers for portlet content, accepts provider responses, and assembles the requested page in the specified page layout.

This command updates the properties in the appConfig.xml file, the configuration file that is used by the Portal mid-tier repository servlet. This configuration file is located in the $MWHOME/user_projects/domains/AllClassicDomain/servers/WLS_PORTAL/stage/portal/portal/configuration/ directory.

12.2.2.2 Syntax
configurePortalPageEngine({encrypt_key}, {resource_url_key}, {use_port}, {use_scheme}, {x509certfile})

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>encrypt_key</td>
<td>Optional. Specifies the HMCA key to obscure the headers used for caching using WebCache. This allows for a more secure cache key, and makes retrieving a cached object by unwanted requests more difficult.</td>
</tr>
<tr>
<td>resource_url_key</td>
<td>Optional. This key, used by the PPE servlet, calculates checksums for URLs that are requested by WSRP and JPDK resource proxying. For WSRP resource proxying to work, the key must be set to an alpha-numeric value of 10 characters or more. In addition, for JPDK proxying, a JNDI environment variable, also called resourceUrlKey, must be set for the provider.</td>
</tr>
<tr>
<td>use_port</td>
<td>Optional. Overrides the port used when the PPE makes requests to the portal. The default, if not specified, is to always use the page request port. Note that if you set useScheme, you must also set the usePort argument. This may be used for other reasons, but mostly it is used when SSL is running between the browser and the PPE but not between the PPE and Portal. In this case, the non-SSL port for loop back requests will be different from the SSL port used by the browser.</td>
</tr>
<tr>
<td>use_scheme</td>
<td>Optional. Overrides the scheme (HTTP or HTTPS) used when the PPE makes requests to the Portal. The default, if not specified, is to always use the page request scheme. Note that if you set useScheme, you must also set the usePort argument.</td>
</tr>
<tr>
<td>x509certfile</td>
<td>Optional. Specifies a file containing a list of certificates to be implicitly trusted by HTTPClient. These certificates are added as trust points to all connections made by HTTPClient using SSL.</td>
</tr>
</tbody>
</table>

12.2.2.3 Example
The following example updates the Portal page engine based on the specified arguments.

```
configurePortalPageEngine(encrypt_key='encryption key', resource_url_key='foo.oracle.com', use_port=9999, use_scheme='page_engine_1', x509certfile='file')
```
12.2.3 listPortalWebcacheConfigAttributes

Command Category: Configuration Commands
Use with WLST: Online

12.2.3.1 Description
Lists the attributes of WebCache configuration used by the Portal repository.

12.2.3.2 Syntax

listPortalWebcacheConfigAttributes ([dad_name])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>dad_name</td>
<td>Optional. Name of the Database Access Descriptor. Default DAD name is 'portal'.</td>
</tr>
</tbody>
</table>

12.2.3.3 Example

The following example lists the WebCache configuration used by the Portal repository. The WebCache host name to which the invalidation messages are sent, the invalidation user name, password and the invalidation port to which the invalidation messages are sent are listed.

```
listPortalWebcacheConfigAttributes(dad_name='portal1')
listPortalWebcacheConfigAttributes('portal1')
----------------
WebCacheConfig
----------------
WebCache Host: foo.oracle.com
WebCache Invalidation Password: invalidator
WebCache Invalidation Port: 6523
WebCache Invalidation User: invalidator
```

12.2.4 listPortalSiteConfigAttributes

Command Category: Configuration Commands
Use with WLST: Online

12.2.4.1 Description
Lists the attributes of the Portal site configuration.

12.2.4.2 Syntax

listPortalSiteConfigAttributes ([dad_name])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>dad_name</td>
<td>Optional. Name of the Database Access Descriptor. Default DAD name is 'portal'.</td>
</tr>
</tbody>
</table>

12.2.4.3 Example

The following example lists the Portal site configuration. Site protocol can be true or false. HTTP is the protocol when site protocol is false and HTTPS is the protocol when the site protocol is true. The site host name and port number are also listed.

```
listPortalSiteConfigAttributes(dad_name='portal1')
```
listPortalSiteConfigAttributes('portal1')

---------------
SiteConfig
---------------
Site Protocol: false
Site Host: foo.oracle.com
Site Port: 8090

12.2.5 listPortalOIDConfigAttributes
Command Category: Configuration Commands
Use with WLST: Online

12.2.5.1 Description
Lists the attributes of the Oracle Internet Directory configuration.

12.2.5.2 Syntax
listPortalOIDConfigAttributes ([dad_name])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>dad_name</td>
<td>Optional. Name of the Database Access Descriptor. Default DAD name is 'portal'.</td>
</tr>
</tbody>
</table>

12.2.5.3 Example
The following example lists the Oracle Internet Directory data, which includes the Oracle Internet Directory host name and port number.

listPortalOIDConfigAttributes(dad_name='portal1')
listPortalOIDConfigAttributes('portal1')

---------------
OidConfig
---------------
OID Port: 13060
OID Host: foo.oracle.com

12.2.6 setPortalWebcacheConfig
Command Category: Configuration Commands
Use with WLST: Online

12.2.6.1 Description
WebCache offers caching, page assembly, and compression features. Oracle WebCache accelerates the delivery of both static and dynamic Web content, and provides load balancing and failover features for Oracle Fusion Middleware.

This command updates the WebCache configuration.

12.2.6.2 Syntax
setPortalWebcacheConfig([dad_name], [host], [inv_port], [inv_user], [inv_passwd])
### 12.2.6.3 Example
The following example updates the WebCache configuration based on the specified values.

```
setPortalWebcacheConfig(dad_name='portal1', host='foo.oracle.com', inv_port='6523', inv_user='invalidator', inv_password='invalidator')
```

### 12.2.7 setPortalOIDConfig

**Command Category:** Configuration Commands

**Use with WLST:** Online

#### 12.2.7.1 Description
Updates the attributes of the Oracle Internet Directory configuration.

#### 12.2.7.2 Syntax

```
setPortalOIDConfig ([dad_name], [host], [port], [protocol], [admin_user], [admin_passwd])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>dad_name</td>
<td>Optional. Name of the Database Access Descriptor. Default DAD name is 'portal'.</td>
</tr>
<tr>
<td>host</td>
<td>Optional. Oracle Internet Directory host name.</td>
</tr>
<tr>
<td>port</td>
<td>Optional. Oracle Internet Directory port number.</td>
</tr>
<tr>
<td>protocol</td>
<td>Optional. Oracle Internet Directory protocol.</td>
</tr>
<tr>
<td>admin_user</td>
<td>Optional. Oracle Internet Directory administrator's name.</td>
</tr>
<tr>
<td>admin_passwd</td>
<td>Optional. Oracle Internet Directory administrator's password.</td>
</tr>
</tbody>
</table>

#### 12.2.7.3 Example
The following example updates the OID configuration based on the specified values.

```
setPortalOIDConfig(dad_name='portal1', host='foo.oracle.com', port='13060', protocol=false, admin_user='cn=orcladmin', admin_passwd='oracle1')
```

### 12.2.8 setPortalMidtierConfig

**Command Category:** Configuration Commands
Use with WLST: Online

12.2.8.1 Description
Updates the Portal repository with the latest Portal mid-tier configuration.

12.2.8.2 Syntax

```
setPortalMidtierConfig([dad_name], [ohs_host], [ohs_port], [ohs_protocol],
[webcache_host], [webcache_inv_user], [webcache_inv_port],
[webcache_inv_passwd])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>dad_name</td>
<td>Optional. Name of the Database Access Descriptor. Default DAD name is 'portal'.</td>
</tr>
<tr>
<td>ohs_host</td>
<td>Optional. Oracle HTTP Server host name.</td>
</tr>
<tr>
<td>ohs_port</td>
<td>Optional. Oracle HTTP Server port number.</td>
</tr>
<tr>
<td>ohs_protocol</td>
<td>Optional. Oracle HTTP Server protocol.</td>
</tr>
<tr>
<td>webcache_host</td>
<td>Optional. The name of the WebCache host to which invalidation messages are sent.</td>
</tr>
<tr>
<td>webcache_inv_user</td>
<td>Optional. The WebCache user name used for sending the invalidation messages.</td>
</tr>
<tr>
<td>webcache_inv_port</td>
<td>Optional. The WebCache port number to which invalidation messages are sent.</td>
</tr>
<tr>
<td>webcache_inv_passwd</td>
<td>Optional. WebCache invalidation password.</td>
</tr>
</tbody>
</table>

12.2.8.3 Example
The following example updates the Portal mid-tier configuration based on the specified values.

```
setPortalMidtierConfig(dad_name='portal1', ohs_host='foo.oracle.com',
ohs_port='8090', ohs_protocol=false, webcache_host='foo.oracle.com',
webcache_inv_user='invalidator', webcache_inv_port='6523',
webcache_inv_passwd='invalidator')
```
Java Required Files (JRF) consists of those components not included in the WebLogic Server installation that provide common functionality for Oracle business applications and application frameworks.

It consists of a number of independently developed libraries and applications that are deployed into a common location. The following components are considered part of Java Required Files: 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.

### 13.1 Java Required Files Commands

Use the commands in Table 13–1 to configure a Managed Server or cluster with Java Required Files (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.

**Note:** To use these JRF custom WLST commands, you must invoke the WLST script from the Oracle Common home. See "Using Custom WLST Commands" in the Oracle Fusion Middleware Administrator’s Guide.

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Section 13.1.1, &quot;applyJRF&quot;</td>
<td>Configures a Managed Server or cluster with Java Required Files applications and services.</td>
<td>Online or Offline</td>
</tr>
<tr>
<td>Section 13.1.2, &quot;cloneDeployments&quot;</td>
<td>Copies the applications and services from Managed Server or cluster and applies them to another Managed Server or cluster.</td>
<td>Online or Offline</td>
</tr>
</tbody>
</table>
13.1.1 applyJRF

Use with WLST: Online or Offline

13.1.1.1 Description

Configures a Managed Server or cluster with Java Required Files (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.

13.1.1.2 Syntax

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>target</td>
<td>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.</td>
</tr>
<tr>
<td>domainDir</td>
<td>The absolute path of the WebLogic Server domain.</td>
</tr>
<tr>
<td>shouldUpdateDomain</td>
<td>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.</td>
</tr>
</tbody>
</table>

13.1.1.3 Example

The following example configures the Managed Server server1 with JRF:

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

13.1.2 cloneDeployments

Use with WLST: Online or Offline

13.1.2.1 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.

13.1.2.2 Syntax

cloneDeployments(domain, source, target, [shouldUpdateDomain])
13.1.2.3 Example

The following example replicates the deployments from sourceServer to destinationServer:

```
wls:/offline> cloneDeployments( '/my_path/user_templates/domains/my_domain', 'sourceServer','destinationServer', 'false')
```
Web Services Custom WLST Commands

The following sections describe the WebLogic Scripting Tool (WLST) commands for Oracle Fusion Middleware Infrastructure Web services, which includes SOA composites, ADF Business Components, and WebCenter services. You can use these commands to manage Web services from the command line.

Topics in this chapter include:
- Section 14.1, "Overview of Web Services WLST Commands"
- Section 14.2, "Web Service and Client Management Commands"
- Section 14.3, "Policy Management Commands"
- Section 14.4, "Policy Repository Upgrade Commands"

For additional details about using these WLST commands for Web services, see the Security and Administrator’s Guide for Web Services.

---

**Note:** 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 Oracle Fusion Middleware Administrator’s Guide.

14.1 Overview of Web Services WLST Commands

You can use WLST, in online mode, to perform Web services configuration and Oracle WSM policy management tasks. You can also use WLST commands to upgrade the Oracle Metadata Services (MDS) repository with new predefined policies with each release.

The Web services WLST commands manage deployed, active, and running Web services applications. They can be executed everywhere in WLST online mode, for example:

```
wlst:/domain/serverConfig
wlst:/domain/domainRuntime
```

The Web services WLST configuration and policy management commands perform many of the same management functions that you can complete using Fusion Middleware Control. When using the WLST commands to manage a Web service of an ADF or WebCenter application, you can apply the change only to a Web service deployed in an application on a specific server. If the application is deployed in a cluster or multi-server environment, you need to make the same change to each of the servers to which the application is deployed. Additionally, when you set or change an
attached policy in ADF and WebCenter Web service and client applications, you must restart the application for the changes to take effect.

In contrast, if you are using the WLST commands to manage a SOA composite, you only need to issue the command once, and the change is propagated to all the server instances in the composite. When you set or change an attached policy in a SOA composite, you do not need to restart it. The SOA fabric runtime engine internally implements all of the policy management changes.

14.1.1 Specifying Application and Composite 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.

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:

```
/soainfra/AdminServer/HelloWorld#1_0
/soainfra/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.

Oracle Infrastructure Web Services client applications are deployed directly to WebLogic Server server instances. Each client 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
```

To specify a SOA composite in a WLST command, use the following format:

```
[zone/]composite[version]
```

The following example shows the sample format for a SOA composite application name:

```
soa_zone1/myComposite[1.0]
```

14.1.2 Web Services WLST Command Categories

Web services WLST commands are divided into the categories described in Table 14–1.
14.2 Web Service and Client Management Commands

Use the WLST commands listed in Table 14–2 to view and manage Web services for deployed, active, and running Web service applications.

<table>
<thead>
<tr>
<th>Table 14–1  Web Services WLST Command Categories</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Category</td>
</tr>
<tr>
<td>Section 14.3, “Policy Management Commands”</td>
</tr>
<tr>
<td>Section 14.4, “Policy Repository Upgrade Commands”</td>
</tr>
</tbody>
</table>

14.2.1 listWebServices

Command Category: Web Service and Client Management

Use with WLST: online
14.2.1.1 Description
Lists the Web service information for an application, SOA composite, or domain. If you don’t 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 using the `detail` argument.

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

```
/domain/server/application#version_number:
   serviceName=service, moduleType=web, moduleName=helloModule
/soainfra/AdminServer/soa-infra:
   serviceName=service, moduleType=soa, compositeName=HelloWorld[1.0]
```

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

14.2.1.3 Examples
The following example lists all the Web services in all applications and composites in the domain. Sample output is also shown in this example.

```
wls:/soainfra/serverConfig> listWebServices()
/soainfra/AdminServer/soa-infra :
   serviceName=service, moduleType=soa, compositeName=HelloWorld[1.0]
   serviceName=bpelprocess1_client_ep, moduleType=soa,
   compositeName=Project1[1.0]
/soainfra/AdminServer/HelloWorld#1_0 :
   serviceName=WssUsernameService, moduleType=web, moduleName=j2wbasicPolicy
```

The following example sets the `detail` argument to `true`. The output from this command provides port and policy details for all applications and composites in the domain. Sample output is also shown in this example.

```
wls:/soainfra/serverConfig> listWebServices(None, None, true)
/soainfra/AdminServer/soa-infra :
   serviceName=service, moduleType=soa, compositeName=HelloWorld[1.0]
   enableTestPage: true
   enableWSDL: true
```

### Argument Definition

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>application</td>
<td>Name and path of the application for which you want to list the Web services. For example, <code>/domain/server/application#version_number</code>. If specified, all Web services in the application are listed.</td>
</tr>
<tr>
<td>composite</td>
<td>Name of the SOA composite for which you want to list the Web services. For example, <code>HelloWorld[1.0]</code>. If specified, all Web services in the composite are listed.</td>
</tr>
</tbody>
</table>
| detail | Optional. Specifies whether to list port and policy details for the Web service. 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. |
14.2.2 listWebServicePorts

Command Category: Web Service and Client Management
Use with WLST: online

14.2.2.1 Description
List 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
```

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>application</td>
<td>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.</td>
</tr>
<tr>
<td>moduleOrCompName</td>
<td>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 and the moduleType argument must be set to soa.</td>
</tr>
</tbody>
</table>
| moduleType      | Module type. Valid options: 
|                 | - web—Use with Web modules (including EJB Web services.)
|                 | - soa—Required for a SOA composite. |
| serviceName     | Name of the Web service in the application or SOA composite for which you want to list the port information. |
14.2.2.3 Example
The following example lists the Web service ports and endpoint URLs for the j2wbasicPolicy service in the soainfra/AdminServer/HelloWorld#1_0 application. Note that the WssUsernameService module name is specified, and the moduleType is set to web.

wls:/soainfra/serverConfig> listWebServicePorts
('/soainfra/AdminServer/HelloWorld#1_0', 'WssUsernameService','web','j2wbasicPolicy')

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

14.2.3 listWebServiceConfiguration
Command Category: Web Service and Client Management
Use with WLST: online

14.2.3.1 Description
List the Web service port configuration for a Web service application or SOA composite.

The output will display the configuration information for the Web service port. For example:

enableREST: false
maxRequestSize: -1

14.2.3.2 Syntax
listWebServiceConfiguration{application,moduleOrCompName,moduleType,serviceName, [subjectName]}

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>application</td>
<td>Name and path of the application for which you want to list the Web services port configuration. For example, /domain/server/application#version_number. To list the port configuration for a Web service application, this argument is required.</td>
</tr>
<tr>
<td>moduleOrCompName</td>
<td>Name of the Web module or SOA composite (for example HelloWorld[1.0]) for which you want to list the Web services port configuration. To list the port configuration for a SOA composite, the composite name is required and the moduleType argument must be set to soa.</td>
</tr>
</tbody>
</table>
| moduleType       | Module type. Valid options are:  
|                  | - web—Use with Web modules (including EJB Web services.).  
|                  | - soa—Required for a SOA composite. |
| serviceName      | Name of the Web service in the application or SOA composite for which you want to list the port configuration. |
| subjectName      | Optional. Policy subject, port, or operation for which you want to list configuration information. |
14.2.3.3 Example
The following example lists the Web service and port configuration information for the application HelloWorld#1_0 for the server soa1 in the domain soainfra. In this example, the Web module name is j2wbasicPolicy, the service name is WssUsernameService, and the subject is a port named JRFWssUsernamePort.

```java
wls:/wls-domain/serverConfig>
listWebServiceConfiguration
('/soainfra/soa1/HelloWorld#1_0','j2wbasicPolicy','web',
'WssUsernameService','JRFWssUsernamePort')
```

14.2.4 setWebServiceConfiguration

Command Category: Web Service and Client Management
Use with WLST: online

14.2.4.1 Description
Set or change the Web service port configuration for a Web service application or SOA composite.

14.2.4.2 Syntax

```java
setWebServiceConfiguration(application,moduleOrCompName,moduleType,serviceName,
subjectName,itemProperties)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>application</td>
<td>Name and path of the application for which you want to set or change the Web services port configuration. For example, /domain/server/application#version_number. To set or change the port configuration for a Web service application, this argument is required.</td>
</tr>
<tr>
<td>moduleOrCompName</td>
<td>Name of the Web module or SOA composite (for example HelloWorld[1.0]) for which you want to set or change the Web services port configuration. To set or change the port configuration for a SOA composite, the composite name is required and the moduleType argument must be set to soa.</td>
</tr>
<tr>
<td>moduleType</td>
<td>Module type. Valid options are:</td>
</tr>
<tr>
<td></td>
<td>- web—Use with Web modules (including EJB Web services.).</td>
</tr>
<tr>
<td></td>
<td>- soa—Required for a SOA composite.</td>
</tr>
<tr>
<td>serviceName</td>
<td>Name of the Web service in the application or SOA composite for which you want to set or change the port configuration.</td>
</tr>
<tr>
<td>subjectName</td>
<td>Policy subject, port or operation name for which you want to set or change the configuration information.</td>
</tr>
</tbody>
</table>
14.2.4.3 Example
The following example enables the port JRFWssUsernamePort for the service WssUsernameService in the Web module j2wbasicPolicy. The service is in the application HelloWorld#1_0 for the server soa1 in the domain soainfra.

```
setWebServiceConfiguration('/soainfra/soa1/HelloWorld#1_0','j2wbasicPolicy','web','WssUsernameService','JRFWssUsernamePort',[['enable', 'true']])
```

14.2.5 listWebServiceClients
Command Category: Web Service and Client Management
Use with WLST: online

14.2.5.1 Description
List 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. You can specify the amount of information to be displayed using the optional detail argument.

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

```
/domin/server/application#version_number:
   serviceRefName=Client, moduleType=wsconn, moduleName=ClientModule
/soainfra/AdminServer/soa-infra:
   serviceRefName=client, moduleType=soa, compositeName=HelloWorld[1.0]
```

14.2.5.2 Syntax
listWebServiceClients{application, composite,[detail]}
14.2.5.3 Examples

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

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

The following example lists the Web service clients for the application `jwsclient_1#1.10` for the server `soa1` in the domain `soainfra`.

```
wlst:/wls-domain/serverConfig>listWebServiceClients('soainfra/soa1/jwsclient_1#1.10')
```

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

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

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

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

14.2.6 listWebServiceClientPorts

Command Category: Web Service and Client Management

Use with WLST: online

14.2.6.1 Description

List 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
```

14.2.6.2 Syntax

```
listWebServiceClientPorts(application,moduleOrCompName,moduleType,serviceRefName)
```
### 14.2.6.3 Examples

The following example lists the client ports for the WssUsernameClient Web module in the /soainfra/soa1/jwsclient_1#1.1.0 application. Note that the moduleType is set to wsconn, and the serviceRefName is set to WssUsernameClient.

```
wlst:/soainfra/serverConfig> listWebServiceClientPorts
(‘/soainfra/soa1/jwsclient_1#1.1.0’,’WssUsernameClient’,’wsconn’, ‘WssUsernameClient’)  
```

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

```
wls:/soainfra/serverConfig> listWebServiceClientPorts(None, ‘HelloWorld[1.0]’,’soa’,’client’)  
```

### 14.2.7 listWebServiceClientStubProperties

#### Command Category: Web Service and Client Management

Use with WLST: online

#### 14.2.7.1 Description

List Web service client port stub properties for an application or SOA composite.

#### 14.2.7.2 Syntax

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

The following example lists the client port stub properties for the JRFWssUsernamePort port of the WssUsernameClient Web module in the /soainfra/soa1/jwsclient_1#1.1.0 application. Note that the moduleType is set to wsconn, and the serviceRefName is set to WssUsernameClient.

```
wlst:/soainfra/serverConfig>listWebServiceClientStubProperties
('/soainfra/soa1/jwsclient_1#1.1.0','WssUsernameClient','wsconn','WssUsernameClient','JRFWssUsernamePort')
```

14.2.8 setWebServiceClientStubProperty

Command Category: Web Service and Client Management

Use with WLST: online

14.2.8.1 Description

Set, change, or delete a single stub property of a Web service client port for an application or SOA composite.

14.2.8.2 Syntax

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>application</td>
<td>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</td>
</tr>
<tr>
<td></td>
<td>To set a client port stub property for an application, this argument is required.</td>
</tr>
<tr>
<td>moduleOrCompName</td>
<td>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 properties.</td>
</tr>
<tr>
<td></td>
<td>To list the client port stub properties information for an SOA composite, the composite name is required and the moduleType argument must be set to soa.</td>
</tr>
<tr>
<td>moduleType</td>
<td>Module type. Valid options are:</td>
</tr>
<tr>
<td></td>
<td>■ web—Use with asynchronous Web service callback client.</td>
</tr>
<tr>
<td></td>
<td>■ soa—Required for a SOA composite.</td>
</tr>
<tr>
<td></td>
<td>■ wsconn—Use with ADF Web service data control or WebCenter portlets.</td>
</tr>
<tr>
<td>serviceRefName</td>
<td>Service reference name of the application or SOA composite for which you want to set the Web service client port stub properties.</td>
</tr>
<tr>
<td>portInfoName</td>
<td>The name of the client port for which you want to list the stub properties.</td>
</tr>
<tr>
<td>propName</td>
<td>Property name of the client port stub property to set, change, or delete.</td>
</tr>
<tr>
<td>propValue</td>
<td>Optional value for the property.</td>
</tr>
</tbody>
</table>

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
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.
To list the client port stub properties information for an SOA composite, the composite name is required and the moduleType argument must be set to soa.

moduleType

Module type. Valid options are:

■ web—Use with asynchronous Web service callback client.
■ soa—Required for a SOA composite.
■ wsconn—Use with ADF Web service data control or WebCenter portlets.

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.

14.2.7.3 Example

The following example lists the client port stub properties for the JRFWssUsernamePort port of the WssUsernameClient Web module in the /soainfra/soa1/jwsclient_1#1.1.0 application. Note that the moduleType is set to wsconn, and the serviceRefName is set to WssUsernameClient.

```
wlst:/soainfra/serverConfig>listWebServiceClientStubProperties
('/soainfra/soa1/jwsclient_1#1.1.0','WssUsernameClient','wsconn','WssUsernameClient','JRFWssUsernamePort')
```
14.2.8.3 Example

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

```
setWebServiceClientStubProperty
('/soainfra/soal/jwsclient_1#1.1.0','WssUsernameClient','wsconn','JRFWssUsernamePort','keystore.recipient.alias','oracle')
```

### 14.2.9 setWebServiceClientStubProperties

**Command Category:** Web Service and Client Management

Use with WLST: online

#### 14.2.9.1 Description

Configure 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 Oracle WSM 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.

#### 14.2.9.2 Syntax

```
setWebServiceClientStubProperties(application, moduleOrCompName, moduleType, serviceRefName, portInfoName, propName)
```
### Argument Definition

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>application</td>
<td>Name and path of the application for which you want to reset the Web services client port stub properties. For example, <code>/domain/server/application#version_number</code> To configure or reset the client port stub properties for an application, this argument is required.</td>
</tr>
<tr>
<td>moduleOrCompName</td>
<td>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 and the moduleType argument must be set to soa.</td>
</tr>
</tbody>
</table>
| moduleType       | Module type. Valid options are:  
- web—Use with asynchronous Web service callback client.  
- soa—Required for a SOA composite.  
- wsconn—Use with ADF Web service data control or WebCenter portlets. |
| 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. |
| 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"],  
["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 |

### 14.2.9.3 Example

The following example resets the client port stub properties ROLE and myprop to ADMIN and myval, 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 /soainfra/soal/jwsclient_1#1.1.0 application. Note that the moduleType is set to wsconn, and the serviceRefName is set to WssUsernameClient.

```wlsсал/sоainfra/serverConfig>setWebServiceClientStubProperties('/soainfra/soal/jwsclient_1#1.1.0',
```
14.3 Policy Management Commands

Use the WLST commands listed in Table 14–3 to manage Oracle WSM Web service and client policies.

When you set or change an attached policy in ADF and WebCenter Web service and client applications, you must restart the application for the changes to take effect. After the policy change is completed, a reminder message is displayed prompting you to restart the application. You can stop and restart the application using the standard stopApplication and startApplication WLST commands. For more information about these commands, see “Deployment Commands” on page 3-18.

Table 14–3 Web Services WLST Policy Management Commands

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>listAvailableWebServicePolicies</td>
<td>Display a list of all the available Oracle Web Services Manager (WSM) policies by category or subject type.</td>
<td>online</td>
</tr>
<tr>
<td>listWebServicePolicies</td>
<td>List Web service port policy information for a Web service in an application or SOA composite.</td>
<td>online</td>
</tr>
<tr>
<td>attachWebServicePolicy</td>
<td>Attach a policy to a Web service port of an application or SOA composite.</td>
<td>online</td>
</tr>
<tr>
<td>attachWebServicePolicies</td>
<td>Attach multiple policies to a Web service port of an application or SOA composite.</td>
<td>online</td>
</tr>
<tr>
<td>enableWebServicePolicy</td>
<td>Enable or disable a policy attached to a port of a Web service application or SOA composite.</td>
<td>online</td>
</tr>
<tr>
<td>enableWebServicePolicies</td>
<td>Enable or disable multiple policies attached to a port of a Web service application or SOA composite.</td>
<td>online</td>
</tr>
<tr>
<td>detachWebServicePolicy</td>
<td>Detach an Oracle WSM policy from a Web service port of an application or SOA composite.</td>
<td>online</td>
</tr>
<tr>
<td>detachWebServicePolicies</td>
<td>Detach multiple Oracle WSM policies from a Web service port of an application or SOA composite.</td>
<td>online</td>
</tr>
<tr>
<td>listWebServiceClientPolicies</td>
<td>List Web service client port policies information for an application or SOA composite.</td>
<td>online</td>
</tr>
<tr>
<td>attachWebServiceClientPolicy</td>
<td>Attach an Oracle WSM policy to a Web service client port of an application or SOA composite.</td>
<td>online</td>
</tr>
<tr>
<td>attachWebServiceClientPolicies</td>
<td>Attach multiple policies to a Web service client port of an application or SOA composite.</td>
<td>online</td>
</tr>
<tr>
<td>enableWebServiceClientPolicy</td>
<td>Enable or disable a policy of a Web service client port of an application or SOA composite.</td>
<td>online</td>
</tr>
</tbody>
</table>
14.3.1 listAvailableWebServicePolicies

Command Category: Policy Management

Use with WLST: online

14.3.1.1 Description
Display a list of all the available Oracle Web Services Manager (WSM) policies by category or subject type.

14.3.1.2 Syntax
listAvailableWebServicePolicies([category],[subject])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>category</td>
<td>Optional. The policy category, for example: 'security', 'management'.</td>
</tr>
<tr>
<td>subject</td>
<td>Optional. The policy subject type, for example: 'server' or 'client'.</td>
</tr>
</tbody>
</table>

14.3.1.3 Example
The following example lists all the available Oracle WSM server security policies in the domain.

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

14.3.2 listWebServicePolicies

Command Category: Policy Management

Use with WLST: online

14.3.2.1 Description
List Web service port policy information for a Web service in an application or SOA composite.

The output will display the Web service port name, the OWSM policies it has attached to it, and if applicable, any policy override properties. For example:

HelloWorldPort:
security: oracle/wss_username_token_service_policy, enabled=true

14.3.2.2 Syntax
listWebServicePolicies(application,moduleOrCompName,moduleType,serviceName,subjectName)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>application</td>
<td>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.</td>
</tr>
<tr>
<td>moduleOrCompName</td>
<td>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 and the moduleType argument must be set to soa.</td>
</tr>
<tr>
<td>moduleType</td>
<td>Module type. Valid options are:</td>
</tr>
<tr>
<td></td>
<td>■ web—Use with a Web service application (including EJB Web services).</td>
</tr>
<tr>
<td></td>
<td>■ soa—Required for a SOA composite.</td>
</tr>
<tr>
<td>serviceName</td>
<td>Name of the Web service in the application or SOA composite for which you want to list the port policy information.</td>
</tr>
<tr>
<td>subjectName</td>
<td>Policy subject, port, or operation name.</td>
</tr>
</tbody>
</table>

14.3.2.3 Examples
The following example lists the Web service port policy information for the application HelloWorld#1_0 for the server soa1 in the domain soainfra. In this example, the Web module name is j2wbasicPolicy, the service name is WssUsernameService, and the subject is a port named JRFWssUsernamePort.

`wls:/wls-domain/serverConfig>listWebServicePolicies('/soainfra/soa1/HelloWorld#1_0','j2wbasicPolicy','web','WssUsernameService','JRFWssUsernamePort')`

The following example lists the port policy information for the SOA composite 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.

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

14.3.3 attachWebServicePolicy
Command Category: Policy Management
Use with WLST: online

14.3.3.1 Description
Attach a policy to a Web service port of an application or SOA composite.
The policyURI is validated through the Oracle WSM Policy Manager APIs if the wsm-pm application is installed on WebLogic Server and is available. 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. For ADF and WebCenter applications, a message is displayed to remind you to restart your application.

---

### 14.3.3.2 Syntax

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>application</td>
<td>Name and path of the application to which you want to attach a Web service policy. For example, /domain/server/application#version_number</td>
</tr>
<tr>
<td>moduleOrCompName</td>
<td>Name of the Web module or SOA composite (for example HelloWorld[1.0]) to which you want to attach a Web service policy.</td>
</tr>
<tr>
<td>moduleType</td>
<td>Module type. Valid options are:</td>
</tr>
<tr>
<td></td>
<td>- web—Use with a Web service application (including EJB Web services).</td>
</tr>
<tr>
<td></td>
<td>- soa—Required for a SOA composite.</td>
</tr>
<tr>
<td>serviceName</td>
<td>Name of the Web service in the application or SOA composite.</td>
</tr>
<tr>
<td>subjectName</td>
<td>Name of the policy subject, port, or operation.</td>
</tr>
<tr>
<td>policyURI</td>
<td>Oracle WSM policy name URI, for example 'oracle/log_policy'</td>
</tr>
<tr>
<td>subjectType</td>
<td>Optional. Policy subject type. Valid options are:</td>
</tr>
<tr>
<td></td>
<td>- P—Port. The default is P.</td>
</tr>
<tr>
<td></td>
<td>- O—Not supported in this release.</td>
</tr>
</tbody>
</table>

---

### 14.3.3.3 Examples

The following example attaches the policy `oracle/wss_username_token_service_policy` to the port `JRFWssUsernamePort` of the Web module `WssUsernameService`. The Web service is part of the application `HelloWorld#1_0` for the server `soa1` in the domain `soainfra`.

```
wls:/wlsl-domain/serverConfig> attachWebServicePolicy
('/soainfra/soa1/HelloWorld#1_0','j2wbasicPolicy','web',
'WssUsernameService','JRFWssUsernamePort','oracle/wss_username_token_service_policy')
```
The following example attaches the policy oracle/log_policy to the port HelloWorld_pt of the service HelloService in the SOA composite HelloWorld[1.0].

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

### 14.3.4 attachWebServicePolicies

Command Category: Policy Management

Use with WLST: online

**14.3.4.1 Description**

Attach multiple policies to a Web service port of an application or SOA composite.

The policyURIs are validated through the Oracle WSM Policy Manager APIs if the wsm-pm application is installed on WebLogic Server and is available. 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. For ADF and WebCenter applications, a message is displayed to remind you to restart your application.

**14.3.4.2 Syntax**

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>application</td>
<td>Name and path of the application to which you want to attach the Web service policies. For example,</td>
</tr>
<tr>
<td></td>
<td>/domain/server/application#version_number</td>
</tr>
<tr>
<td></td>
<td>To attach the policies to a port of a Web service application, this argument is required.</td>
</tr>
<tr>
<td>moduleOrCompName</td>
<td>Name of the Web module or SOA composite (for example&gt;HelloWorld[1.0]) to which you want to attach Web service</td>
</tr>
<tr>
<td></td>
<td>policies.</td>
</tr>
<tr>
<td></td>
<td>To attach the policies to a port of a SOA composite, the composite name is required and the moduleType argument</td>
</tr>
<tr>
<td></td>
<td>must be set to soa.</td>
</tr>
<tr>
<td>moduleType</td>
<td>Module type. Valid options are:</td>
</tr>
<tr>
<td></td>
<td>■ web—Use with a Web service application (including EJB Web services).</td>
</tr>
<tr>
<td></td>
<td>■ soa—Required for a SOA composite.</td>
</tr>
<tr>
<td>serviceName</td>
<td>Name of the Web service in the application or SOA composite.</td>
</tr>
<tr>
<td>subjectName</td>
<td>Name of the policy subject, port, or operation.</td>
</tr>
</tbody>
</table>
14.3.4.3 Example
The following example attaches the policies "oracle/log_policy", "oracle/wss_username_token_service_policy" to the port JRFWssUsernamePort of the Web module WssUsernameService. The Web service is part of the application HelloWorld#1_0 for the server soa1 in the domain soainfra.

```
wls:/wls-domain/serverConfig>attachWebServicePolicies
'/soainfra/soa/HelloWorld#1_0','j2wbasicPolicy','web',
'WssUsernameService','JRFWssUsernamePort',
["oracle/log_policy", "oracle/wss_username_token_service_policy")
```

14.3.5 enableWebServicePolicy
Command Category: Policy Management
Use with WLST: online

14.3.5.1 Description
Enable or disable 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. For ADF and WebCenter applications, a message is displayed to remind you to restart your application.

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
</table>
| policyURIs   | List of Oracle WSM 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:                          |
|              | P—Port. The default is P.                                                  |
|              | O—Not supported in this release.                                           |

**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.
14.3.5.3 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 soa1 in the domain soainfra.

```
  wls:/wls-domain/serverConfig>enableWebServicePolicy
  ('/soainfra/soa1/HelloWorld#1_0','j2wbasicPolicy','web','WssUsernameService','JRFWssUsernamePort','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 HelloWorld[1.0].

```
  wls:/wls-domain/serverConfig>enableWebServicePolicy(None, '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 HelloWorld[1.0].

```
  wls:/wls-domain/serverConfig>enableWebServicePolicy(None, 'HelloWorld[1.0]',
  'soa','HelloService','HelloWorld_pt','oracle/log_policy',false)
```
14.3.6 enableWebServicePolicies

Command Category: Policy Management

Use with WLST: online

14.3.6.1 Description

Enable or disable 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. For ADF and WebCenter applications, a message is displayed to remind you to restart your application.

---

14.3.6.2 Syntax

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

---

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>application</td>
<td>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.</td>
</tr>
<tr>
<td>moduleOrCompName</td>
<td>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 and the moduleType argument must be set to soa.</td>
</tr>
<tr>
<td>moduleType</td>
<td>Module type. Valid options are: ■ web—Use with a Web service application (including EJB Web services). ■ soa—Required for a SOA composite.</td>
</tr>
<tr>
<td>serviceName</td>
<td>Name of the Web service in the application or SOA composite.</td>
</tr>
<tr>
<td>subjectName</td>
<td>Name of the policy subject, port, or operation.</td>
</tr>
<tr>
<td>policyURIs</td>
<td>List of Oracle WSM policy name URIs, for example [&quot;oracle/log_policy&quot;,&quot;oracle/wss_username_token_service_policy&quot;] If the policyURIs that you specify are not attached, an error message is displayed and/or an exception is thrown.</td>
</tr>
<tr>
<td>enable</td>
<td>Optional. Specifies whether to enable or disable the policies. Valid options are: ■ true—Enables the policies. The default is true. ■ false—Disables the policies. If you omit this argument, the policies are enabled.</td>
</tr>
</tbody>
</table>
Example

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 soa1 in the domain soainfra.

```
    wls:/wls-domain/serverConfig>enableWebServicePolicy
    ('/soainfra/soa1/HelloWorld#1_0','j2wbasicPolicy','web',
    'WssUsernameService','JRFWssUsernamePort',
    "oracle/log_policy", "oracle/wss_username_token_service_policy"],true)
```

### 14.3.7 detachWebServicePolicy

Command Category: Policy Management

Use with WLST: online

#### 14.3.7.1 Description

Detach an Oracle WSM 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. For ADF and WebCenter applications, a message is displayed to remind you to restart your application.

#### 14.3.7.2 Syntax

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>application</td>
<td>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.</td>
</tr>
<tr>
<td>moduleOrCompName</td>
<td>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 and the moduleType argument must be set to soa.</td>
</tr>
</tbody>
</table>
14.3.7.3 Examples

The following example detaches the policy oracle/wss_username_token_service_policy from the port JRFWssUsernamePort of the Web module WssUsernameService. The Web service is part of the application HelloWorld#1_0 for the server soa1 in the domain soainfra.

```
> detachWebServicePolicy('/soainfra/soa1/HelloWorld#1_0','j2wbasicPolicy','web','WssUsernameService','JRFWssUsernamePort','oracle/wss_username_token_service_policy')
```

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

```
> detachWebServicePolicy(None, 'HelloWorld[1.0]','soa','HelloService','HelloWorld_pt','oracle/log_policy')
```

14.3.8 detachWebServicePolicies

Command Category: Policy Management

Use with WLST: online

14.3.8.1 Description

Detach multiple Oracle WSM 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. For ADF and WebCenter applications, a message is displayed to remind you to restart your application.
14.3.8.2 Syntax

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>application</td>
<td>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.</td>
</tr>
<tr>
<td>moduleOrCompName</td>
<td>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 and the moduleType argument must be set to soa.</td>
</tr>
</tbody>
</table>
| moduleType        | Module type. Valid options are:  
|                   |  ■ web—Use with a Web service application (including EJB Web services).  
|                   |  ■ soa—Required for a SOA composite. |
| serviceName       | Name of the Web service in the application or SOA composite.                                     |
| subjectName       | Name of the policy subject, port, or operation.                                                   |
| policyURIs        | List of Oracle WSM policy name URIs, for example ['oracle/log_policy', 'oracle/wss_username_token_service_policy']. 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:  
|                   |  ■ P—Port. The default is P.  
|                   |  ■ O—Not supported in this release.                                                               |

14.3.8.3 Example

The following example detaches the policies "oracle/log_policy", "oracle/wss_username_token_service_policy" from the port JRFWssUsernamePort of the Web module WssUsernameService of the Web service is part of the application HelloWorld#1.0 for the server soa1 in the domain soainfra.

wls:/wls-domain/serverConfig> detachWebServicePolicies
('soainfra/soa1/HelloWorld#1_0', 'j2wbasicPolicy', 'web', 'WssUsernameService', 'JRFWssUsernamePort', ['oracle/log_policy', 'oracle/wss_username_token_service_policy'])

14.3.9 listWebServiceClientPolicies

Command Category: Policy Management

Use with WLST: online

14.3.9.1 Description

List Web service client port policies information for an application or SOA composite.
The output will display the Web service client/reference port name and the Oracle WSM policies it has attached to it. For example:

```
test-port:
security: oracle/wss_username_token_client_policy, enabled=true
```

### 14.3.9.2 Syntax

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>application</code></td>
<td>Name and path of the application for which you want to list the Web service client port policy information. For example, <code>/domain/server/application#version_number</code></td>
</tr>
</tbody>
</table>
| `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 client port policy information for a SOA composite, the composite name is required and the `moduleType` argument must be set to `soa`.
| `moduleType`     | Module type. Valid options are:
|                  | ■ `web`—Use with asynchronous Web service callback client.
|                  | ■ `soa`—Required for a SOA composite.
|                  | ■ `wsconn`—Use with ADF Web service data control and WebCenter portlets.
| `serviceRefName` | The service reference name of the application or composite. |
| `portInfoName`   | The client port name. |

### 14.3.9.3 Example

The following example lists the Web service client port policy information for the application `jwsclient_1#1.1.0` for the server `soa1` in the domain `soainfra`. 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('/soainfra/soa1/jwsclient_1#1.1.0','WssUsernameClient','wsconn','WssUsernameClient','JRFWssUsernamePort')
```

### 14.3.10 attachWebServiceClientPolicy

Command Category: Policy Management

Use with WLST: online

#### 14.3.10.1 Description

Attach a Oracle WSM policy to a Web service client port of an application or SOA composite.

The `policyURI` is validated through the Oracle WSM Policy Manager APIs if the `wsm-pm` application is installed on WebLogic Server and is available. 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. For ADF and WebCenter applications, a message is displayed to remind you to restart your application.

14.3.10.2 Syntax

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>application</td>
<td>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.</td>
</tr>
<tr>
<td>moduleOrCompName</td>
<td>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 and the moduleType argument must be set to soa.</td>
</tr>
<tr>
<td>moduleType</td>
<td>Module type. Valid options are:</td>
</tr>
<tr>
<td></td>
<td>- web—Use with asynchronous Web service callback client.</td>
</tr>
<tr>
<td></td>
<td>- soa—Required for a SOA composite.</td>
</tr>
<tr>
<td></td>
<td>- wsconn—Use with ADF Web service data control and WebCenter portlets.</td>
</tr>
<tr>
<td>serviceRefName</td>
<td>The service reference name of the application or composite.</td>
</tr>
<tr>
<td>portInfoName</td>
<td>The client port to which you want to attach the Oracle WSM client policy.</td>
</tr>
<tr>
<td>policyURI</td>
<td>The Oracle WSM policy name URI, for example oracle/wss_username_token_client_policy&quot;</td>
</tr>
<tr>
<td></td>
<td>If the policy that you specify is already attached or exists, then this command enables the policy if it is disabled.</td>
</tr>
<tr>
<td>subjectType</td>
<td>Optional. Policy subject type. Valid options are:</td>
</tr>
<tr>
<td></td>
<td>- P—Port. The default is P.</td>
</tr>
<tr>
<td></td>
<td>- O—Not supported in this release.</td>
</tr>
</tbody>
</table>

14.3.10.3 Examples

The following example attaches the client policy oracle/wss_username_token_client_policy to the port JRFWssUsernamePort of the Web module WssUsernameClient. The Web service is part of the application jwsclient_1#1.1.0 for the server soal in the domain soainfra.

wls:/wls-domain/serverConfig>attachWebServiceClientPolicy
("/soainfra/soal/jwsclient_1#1.1.0','WssUsernameClient','wsconn',
"oracle/wss_username_token_client_policy",null,null,null,null)
The following example attaches the client policy `oracle/log_policy` to the client port `HelloWorld_pt` in the SOA composite `HelloWorld[1.0]`.

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

### 14.3.11 attachWebServiceClientPolicies

**Command Category:** Policy Management

**Use with WLST:** online

#### 14.3.11.1 Description

Attach multiple policies to a Web service client port of an application or SOA composite.

The policyURIs are validated through the Oracle WSM Policy Manager APIs if the wsm-pm application is installed on WebLogic Server and is available. 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. For ADF and WebCenter applications, a message is displayed to remind you to restart your application.

#### 14.3.11.2 Syntax

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>application</strong></td>
<td>Name and path of the application for which you want to attach Oracle WSM client policies to the Web service client port. For example, <code>/domain/server/application#version_number</code> To attach policies to a client port of a Web service application, this argument is required.</td>
</tr>
<tr>
<td><strong>moduleOrCompName</strong></td>
<td>Name of the Web module or SOA composite (for example <code>HelloWorld[1.0]</code>) 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 and the moduleType argument must be set to <code>soa</code>.</td>
</tr>
</tbody>
</table>
| **moduleType**    | Module type. Valid options are:  
  - `web`—Use with asynchronous Web service callback client.  
  - `soa`—Required for a SOA composite.  
  - `wsconn`—Use with ADF Web service data control and WebCenter portlets. |
14.3.11.3 Examples

The following example attaches the policy oracle/wss_username_token_client_policy to the port JRFWssUsernamePort of the Web module WssUsernameClient. The Web service is part of the application jwsclient_1#1.1.0 for the server soa1 in the domain soainfra.

```
ws1: /wl-domain/serverConfig> attachWebServiceClientPolicy
(/soainfra/soa1/jwsclient_1#1.1.0', 'WssUsernameClient', 'wsconn', 'WssUsernameClient', 'JRFWssUsernamePort', "oracle/wss_username_token_client_policy")
```

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

```
ws1: /wl-domain/serverConfig> attachWebServiceClientPolicy
(None, 'HelloWorld[1.0]', 'soa', 'client', 'HelloWorld_pt', 'oracle/log_policy')
```

14.3.12 enableWebServiceClientPolicy

Command Category: Policy Management

Use with WLST: online

14.3.12.1 Description

Enable or disable 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. For ADF and WebCenter applications, a message is displayed to remind you to restart your application.

14.3.12.2 Syntax

```
enableWebServiceClientPolicy{application, moduleOrCompName, moduleType, serviceRefName, portInfoName, policyURI, [enable], [subjectType=None] }
```
14.3.12.3 Examples

The following example enables the client policy oracle/wss_username_token_client_policy of the port JRFWssUsernamePort of the Web module WssUsernameClient. The Web service is part of the application jwsclient_1#1.1.0 for the server soa1 in the domain soainfra.

```
wlst:/wls-domain/serverConfig>enableWebServiceClientPolicy
('/',soainfra.soal/jwsclient_1#1.1.0', 'WssUsernameClient', 'wsconn', 'WssUsernameClient', 'JRFWssUsernamePort', "oracle/wss_username_token_client_policy",true)
```

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

```
wlst:/wls-domain/serverConfig>enableWebServiceClientPolicy(  None,  '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 HelloWorld[1.0].

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

Command Category: Policy Management
Use with WLST: online

14.3.13.1 Description
Enable or disable 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. For ADF and WebCenter applications, a message is displayed to remind you to restart your application.

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>application</td>
<td>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.</td>
</tr>
<tr>
<td>moduleOrCompName</td>
<td>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 and the moduleType argument must be set to soa.</td>
</tr>
</tbody>
</table>
| moduleType     | Module type. Valid options are:  
- web—Use with asynchronous Web service callback client.  
- soa—Required for a SOA composite.  
- wsconn—Use with ADF Web service data control and WebCenter portlets. |
| serviceRefName | The service reference name of the application or composite.  |
| portInfoName   | The name of the client port to which you want to attach the Oracle WSM client policies.  |
| policyURIs     | The list of Oracle WSM 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:  
- true—Enables the policy. The default is true.  
- false—Disables the policy.  
If you omit this argument, the policies are enabled. |
14.3.13.3 Example

The following example enables the client policies oracle/log_policy and oracle/wss_username_token_client_policy of the port JRFWssUsernamePort of the Web module WssUsernameClient. The Web service is part of the application jwsclient_1#1.1.0 for the server soa1 in the domain soainfra.

```bash
wls:/wls-domain/serverConfig> enableWebServiceClientPolicies
('/soainfra/soa1/jwsclient_1#1.1.0','WssUsernameClient','wsconn','WssUsernameClient','JRFWssUsernamePort',
["oracle/log_policy", "oracle/wss_username_token_client_policy"], true )
```

14.3.14 detachWebServiceClientPolicy

**Command Category:** Policy Management

**Use with WLST:** online

14.3.14.1 Description

Detach 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. For ADF and WebCenter applications, a message is displayed to remind you to restart your application.

14.3.14.2 Syntax

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>application</td>
<td>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</td>
</tr>
<tr>
<td></td>
<td>To detach a policy from a client port of a Web service application, this argument is required.</td>
</tr>
<tr>
<td>moduleOrCompName</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>To detach a policy from a client port of a SOA composite, the composite name is required and the moduleType argument must be set to soa.</td>
</tr>
</tbody>
</table>
The following example detaches the client policy oracle/wss_username_token_client_policy from the port JRFWssUsernamePort of the Web module WssUsernameClient. The Web service is part of the application jwsclient_1#1.1.0 for the server soa1 in the domain soainfra.

```
wlsls:/wls-domain/serverConfig> detachWebServiceClientPolicy
('/soainfra/soa1/jwsclient_1#1.1.0','WssUsernameClient','wsconn','WssUsernameClient','JRFWssUsernamePort','oracle/wss_username_token_client_policy')
```

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

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

## 14.3.15 detachWebServiceClientPolicies

**Command Category:** Policy Management

**Use with WLST:** online

### 14.3.15.1 Description

Detach 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. For ADF and WebCenter applications, a message is displayed to remind you to restart your application.

### 14.3.15.2 Syntax

```
detachWebServiceClientPolicies(application, moduleOrCompName, moduleType,
```

---

**Argument** | **Definition**
--- | ---

moduleType | Module type. Valid options are:

• web—Use with asynchronous Web service callback client.
• soa—Required for a SOA composite.
• wsconn—Use with ADF Web service data control and WebCenter portlets.

serviceRefName | The service reference name of the application or composite.

portInfoName | The client port from which you want to detach the Oracle WSM client policy.

policyURI | The Oracle WSM policy name URI, for example oracle/wss_username_token_client_policy

subjectType | Optional. Policy subject type. Valid options are:

• P—Port. The default is P.
• O—Not supported in this release.
14.3.15.3 Example
The following example detaches the client policies oracle/log_policy and oracle/wss_username_token_client_policy from the port JRFWssUsernamePort of the Web module WssUsernameClient. The Web service is part of the application jwsclient_1#1.1.0 for the server soa1 in the domain soainfra.

```
wlsh:/wls-domain/serverConfig> detachWebServiceClientPolicies
('/soainfra/soa1/jwsclient_1#1.1.0','WssUsernameClient','wsconn','WssUsernameClient','JRFWssUsernamePort',
['oracle/log_policy','oracle/wss_username_token_client_policy'])
```

14.3.16 setWebServicePolicyOverride
Command Category: Policy Management
Use with WLST: online

14.3.16.1 Description
Configure the Web service port policy override properties of an application or SOA composite.
14.3.16.2 Syntax

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>application</td>
<td>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.</td>
</tr>
<tr>
<td>moduleOrCompName</td>
<td>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 and the moduleType argument must be set to soa.</td>
</tr>
</tbody>
</table>
| moduleType             | Module type. Valid options are:  
  - web—Use with a Web service application.  
  - soa—Required for a SOA composite. |
| serviceName            | Name of the Web service in the application or SOA composite. |
| subjectName            | Name of the policy subject, port, or operation. |
| policyURI              | Oracle WSM policy name URI, for example `oracle/log_policy` to which the override properties will be applied. 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:  
  ```plaintext
  [("name","value")]
  ```
  For example: ```plaintext
  [("ROLE","ADMIN"),("myprop","myval")]
  ```
  If this argument is set to None, then all policy overrides are removed. |

14.3.16.3 Examples

The following example configures the override properties for the policy oracle/wss_username_token_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 soa1 in the domain soainfra.

```plaintext
wls:/wls-domain/serverConfig> setWebServicePolicyOverride 
{"soainfra soa1/HelloWorld#1_0", "j2wbasicPolicy", "web", "WssUsernameService", "JRFWssUsernamePort", 
"oracle/wss_username_token_service_policy", 
[("ROLE","ADMIN"),("myprop","myval")]
```

14.4 Policy Repository Upgrade Commands

Use the commands listed in Table 14–4 to update the WSM policies stored in the Oracle MDS repository.

For additional information, see "Upgrading the Oracle WSM Policies in the MDS Repository" the Security and Administrator’s Guide for Web Services.
Additional MDS WLST commands are described in Chapter 9, "Metadata Services (MDS) Custom WLST Commands."

**Table 14–4  Policy Repository Upgrade Commands**

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>upgradeWSMPolicyRepository</td>
<td>Upgrade the Oracle WSM predefined policies stored in the Oracle MDS repository with any new predefined policies that are provided in the latest installation of the Oracle Fusion Middleware software.</td>
<td>online</td>
</tr>
<tr>
<td>resetWSMPolicyRepository</td>
<td>Delete the existing policies stored in the Oracle MDS repository and refresh it with the latest set of predefined policies that are provided in the new installation of the Oracle Fusion Middleware software.</td>
<td>online</td>
</tr>
</tbody>
</table>

### 14.4.1 upgradeWSMPolicyRepository

Command Category: Policy Repository Upgrade

Use with WLST: online

#### 14.4.1.1 Description

Upgrade the Oracle WSM predefined policies stored in the Oracle MDS 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.

#### 14.4.1.2 Syntax

upgradeWSMPolicyRepository()

#### 14.4.1.3 Example

The following example upgrades the existing installation with policies provided in the latest release:

wls:/wls-domain/serverConfig> upgradeWSMPolicyRepository()

### 14.4.2 resetWSMPolicyRepository

Command Category: Policy Repository Upgrade

Use with WLST: online
14.4.2.1 Description
Delete the existing policies stored in the Oracle MDS 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 MDS repository before loading the new predefined policies.

14.4.2.2 Syntax
resetWSMPolicyRepository(clearStore)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>clearStore</td>
<td>Policies to be deleted. Valid values are:</td>
</tr>
<tr>
<td></td>
<td>■ true—All policies in the repository, including custom user policies, are deleted.</td>
</tr>
<tr>
<td></td>
<td>■ false—Only the predefined policies supplied by Oracle are deleted. The default is false.</td>
</tr>
</tbody>
</table>

14.4.2.3 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)
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.

For additional information about using the Diagnostic Framework, see "Diagnosing Problems" in the Oracle Fusion Middleware Administrator’s Guide.

**Table 15–1** lists the different categories of Diagnostic Framework commands.

<table>
<thead>
<tr>
<th>Command Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incident Commands</td>
<td>View problems and incidents and to create incidents.</td>
</tr>
<tr>
<td>Diagnostic Dump Commands</td>
<td>Display information about dumps and to execute dumps.</td>
</tr>
</tbody>
</table>

**15.1 Incident Commands**

Use the commands in **Table 15–2** to view problems and incidents and to create incidents.

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>createIncident</td>
<td>Create a diagnostic incident.</td>
<td>Online</td>
</tr>
<tr>
<td>getIncidentFile</td>
<td>Retrieves the contents of the specified incident file.</td>
<td>Online</td>
</tr>
<tr>
<td>listADR Homes</td>
<td>List the set of ADR Home paths.</td>
<td>Online</td>
</tr>
<tr>
<td>listIncidents</td>
<td>List a set of diagnostic incidents.</td>
<td>Online</td>
</tr>
<tr>
<td>listProblems</td>
<td>List a set of diagnostic problems.</td>
<td>Online</td>
</tr>
<tr>
<td>showIncident</td>
<td>Show the details of a specified incident.</td>
<td>Online</td>
</tr>
</tbody>
</table>
15.1.1 createIncident

Use with WLST: Online

15.1.1.1 Description
Creates a diagnostic incident, using the specified information to determine the set of diagnostic rules and actions to execute.

15.1.1.2 Syntax

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>adrHome</td>
<td>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</td>
</tr>
</tbody>
</table>
| 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.                            |
| appName    | 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. |

15.1.1.3 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: 55  
Problem Id: 4  
Problem Key: MDS-50400 [MANUAL]  
Incident Time: 25th June 2009 11:55:45 GMT  
Error Message Id: MDS-50400  
Flood Controlled: false

15.1.2 getIncidentFile

Use with WLST: Online
15.1.2.1 Description
Retrieves the contents of the specified incident file.

15.1.2.2 Syntax
getIncidentFile(id, name [,outputFile] [,adrHome] [,server])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The ID of the incident that you want to retrieve.</td>
</tr>
<tr>
<td>name</td>
<td>The name of the file to retrieve. To find the name of the file, use the showIncident command.</td>
</tr>
<tr>
<td>outputFile</td>
<td>The name of the file to which to write the output.</td>
</tr>
<tr>
<td>adrHome</td>
<td>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</td>
</tr>
<tr>
<td>server</td>
<td>The name of the Managed Server from which to collect information. This argument is valid only when you are connected to the Administration Server.</td>
</tr>
</tbody>
</table>

15.1.2.3 Example
The following example writes the contents of the incident dms_metrics3_i1.dmp to the specified output file:

```java
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

15.1.3 listADRHome
Use with WLST: Online

15.1.3.1 Description
Lists the paths of all of the ADR Homes for the server.

15.1.3.2 Syntax
listADRHome{server}

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>server</td>
<td>The name of the Managed Server from which to collect information. This argument is valid only when you are connected to the Administration Server.</td>
</tr>
</tbody>
</table>

15.1.3.3 Example
The following example lists the paths of the ADR homes:

```java
listADRHome()
diag/ofm/base_domain/WLS_Spaces
diag/ofm/fusionapps/GeneralLedger
```
15.1.4 listIncidents

Use with WLST: Online

15.1.4.1 Description
Lists the set of diagnostic incidents for the given problem ID, if specified, or all available incidents.

15.1.4.2 Syntax
listIncidents([id],[adrHome])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The ID of the problem for which you want to list the set of diagnostic incidents.</td>
</tr>
<tr>
<td>adrHome</td>
<td>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</td>
</tr>
</tbody>
</table>

15.1.4.3 Example
The following example lists the incidents associated with the problem with the ID 1

```
listIncidents(id="1")
```

<table>
<thead>
<tr>
<th>Incident Id</th>
<th>Problem Key</th>
<th>Incident Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>MDS-50300  [WLS_Spaces]  [oracle.mds.repos]</td>
<td>Mon Jul 13 11:22:12 PDT</td>
</tr>
<tr>
<td>24</td>
<td>MDS-50300  [WLS_Spaces]  [oracle.mds.repos]</td>
<td>Thu Jul 09 15:11:35 PDT</td>
</tr>
</tbody>
</table>

15.1.5 listProblems

Use with WLST: Online

15.1.5.1 Description
Lists the set of diagnostic problems associated with the specified ADR Home.

15.1.5.2 Syntax
listProblems([adrHome],[server])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>adrHome</td>
<td>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</td>
</tr>
<tr>
<td>server</td>
<td>The name of the Managed Server from which to collect information. This argument is valid only when you are connected to the Administration Server.</td>
</tr>
</tbody>
</table>

15.1.5.3 Example
The following example lists the diagnostic problems in the default ADR home:
Diagnostic Dump Commands

15.1.6 showIncident

Use with WLST: Online

15.1.6.1 Description

Shows the details of the specified incident.

15.1.6.2 Syntax

showIncident(id, [adrHome],[server])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The ID of the incident that you want to view.</td>
</tr>
<tr>
<td>adrHome</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>The default ADR Home is the following location:</td>
</tr>
<tr>
<td></td>
<td>ADR_BASE/diag/OFM/domain_name/server_name</td>
</tr>
<tr>
<td>server</td>
<td>The name of the Managed Server from which to collect information. This argument is valid only when you are connected to the Administration Server.</td>
</tr>
</tbody>
</table>

15.1.6.3 Example

The following example displays information about the incident with the ID 10:

showIncident(id="10")

Incident Id: 10
Problem Id: 1
Problem Key: MDS-50300 [WLS_Spaces] [oracle.mds.repos]
Incident Time: 25th June 2009 10:12:15 GMT
Error Message Id: MDS-50300
Execution Context: 0000ICK4rbYC8xT6uBf9EH1AX1gF000000
Flood Controlled: false
Dump Files:
- dms_ecidctxl_i1.dmp
- jvm_threads2_i1.dmp
- dms_metrics3_i1.dmp
- odl_logs4_i1.dmp
- diagnostic_image_AdminServer_2009_06_25_11_12_15.zip
- readme.txt

15.2 Diagnostic Dump Commands

Use the commands in Table 15–3 to display information about dumps and to execute dumps.
15.2.1 describeDump

Use with WLST: Online

15.2.1.1 Description
Displays a description of the specified diagnostic dump.

15.2.1.2 Syntax
```
describeDump(name [,appName] [.server])
```

15.2.1.3 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 commands.
```
describeDump(name="odl.logs")
```

15.2.2 executeDump

Use with WLST: Online

15.2.2.1 Description
Executes the specified diagnostic dump.

15.2.2.2 Syntax
```
executeDump(name [,args] [,outputFile] [,id] [,adrHome] [,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"})

### 15.2.2.3 Examples

The following example executes the dump with the name jvm.threads and writes it to the file dumpout.txt:

```bash
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:

```bash
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`:

```bash
executeDump('java.sysprops', args={"prop" : "os.name"})
```

### 15.2.3 listDumps

Use with WLST: Online

#### 15.2.3.1 Description

Displays the set of diagnostic dumps that can be executed.

#### 15.2.3.2 Syntax

`listDumps([appNane] [,server])`
15.2.3.3 Example

The following example lists all of the available dumps.

```
listDumps()
dms.metrics
dms.classhistogram
dms.threads
odl.logs
```

Use the command `describeDump(name=<dumpName>)` for help on a specific dump.
The following sections describe the Oracle Fusion Middleware Information Rights Management custom WLST commands in detail. Topics include:

- Section 16.1, "Overview of WLST IRM Commands"
- Section 16.2, "General Server Commands"
- Section 16.3, "Migration Commands"
- Section 16.4, "Test Content Commands"
- Section 16.5, "Languages Support Commands"
- Section 16.6, "Oracle IRM Desktop Installers Commands"

For additional information about Oracle Information Rights Management, see Oracle IRM Administrator’s Guide.

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 Oracle Fusion Middleware Administrator’s Guide.

16.1 Overview of WLST IRM Commands

WLST IRM commands are divided into the following categories:

<table>
<thead>
<tr>
<th>Command Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>General Server Commands</td>
<td>Make general changes to Oracle IRM Server settings.</td>
</tr>
<tr>
<td>Migration Commands</td>
<td>Back up and migrate Oracle IRM Server user data.</td>
</tr>
<tr>
<td>Test Content Commands</td>
<td>Set up test content for users of Oracle IRM Desktop.</td>
</tr>
<tr>
<td>Languages Support Commands</td>
<td>Set up languages support for users of Oracle IRM Server.</td>
</tr>
<tr>
<td>Oracle IRM Desktop Installers Commands</td>
<td>Set up software installation support for Oracle IRM Desktop.</td>
</tr>
</tbody>
</table>


16.2 General Server Commands

Use the WLST commands listed in Table 16–2 to make general changes to Oracle IRM Server settings.

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>addIRMRefreshPeriod</td>
<td>Create a new refresh period.</td>
<td>Online</td>
</tr>
<tr>
<td>getIRMRefreshPeriod</td>
<td>Display an existing refresh period.</td>
<td>Online</td>
</tr>
<tr>
<td>getIRMRefreshPeriods</td>
<td>Display all the refresh periods.</td>
<td>Online</td>
</tr>
<tr>
<td>removeIRMRefreshPeriod</td>
<td>Remove an existing refresh period.</td>
<td>Online</td>
</tr>
<tr>
<td>updateIRMRefreshPeriod</td>
<td>Update an existing refresh period.</td>
<td>Online</td>
</tr>
<tr>
<td>addIRMSyncWindow</td>
<td>Create a new sync window.</td>
<td>Online</td>
</tr>
<tr>
<td>getIRMSyncWindow</td>
<td>Display an existing sync window.</td>
<td>Online</td>
</tr>
<tr>
<td>getIRMSyncWindows</td>
<td>Display all the sync windows.</td>
<td>Online</td>
</tr>
<tr>
<td>removeIRMSyncWindow</td>
<td>Remove an existing sync window.</td>
<td>Online</td>
</tr>
<tr>
<td>updateIRMSyncWindow</td>
<td>Update an existing sync window.</td>
<td>Online</td>
</tr>
<tr>
<td>getIRMCryptoSchema</td>
<td>Display the cryptography algorithm.</td>
<td>Online</td>
</tr>
<tr>
<td>setIRMCryptoSchema</td>
<td>Set the cryptography algorithm.</td>
<td>Online</td>
</tr>
<tr>
<td>getIRMDeviceCount</td>
<td>Display the device count.</td>
<td>Online</td>
</tr>
<tr>
<td>setIRMDeviceCount</td>
<td>Set the device count.</td>
<td>Online</td>
</tr>
<tr>
<td>getIRMJournalCleanUp</td>
<td>Display the current report record clean-up values.</td>
<td>Online</td>
</tr>
<tr>
<td>setIRMJournalCleanUp</td>
<td>Set report record clean-up values.</td>
<td>Online</td>
</tr>
<tr>
<td>getIRMLicenseStateCleanUp</td>
<td>Display the license state clean-up frequency.</td>
<td>Online</td>
</tr>
<tr>
<td>setIRMLicenseStateCleanUp</td>
<td>Set the license state clean-up frequency.</td>
<td>Online</td>
</tr>
<tr>
<td>getIRMPrivacyURL</td>
<td>Display the URL of the privacy statement page.</td>
<td>Online</td>
</tr>
<tr>
<td>setIRMPrivacyURL</td>
<td>Set the URL of the privacy statement page.</td>
<td>Online</td>
</tr>
<tr>
<td>getIRMKeyStore</td>
<td>Display the type and location of the Oracle IRM keystore.</td>
<td>Online</td>
</tr>
<tr>
<td>setIRMKeyStore</td>
<td>Set the type and location of the Oracle IRM keystore.</td>
<td>Online</td>
</tr>
</tbody>
</table>

16.2.1 addIRMRefreshPeriod

Online command that creates a new refresh period.

16.2.1.1 Description

This command creates a refresh period. A refresh period is the maximum length of time that a user can use rights before the rights are refreshed from the server.

16.2.1.2 Syntax

addIRMRefreshPeriod(duration,dtype)
16.2.1.3 Examples
The following example creates a refresh period of 5 hours:

```
wls:/base_domain/serverConfig> addIRMRefreshPeriod(5,"HOURS")
```

The following example creates a refresh period of 50 minutes:

```
wls:/base_domain/serverConfig> addIRMRefreshPeriod(50,"MINUTES")
```

16.2.2 getIRMRefreshPeriod
Online command that displays an existing refresh period.

16.2.2.1 Description
This command displays the refresh period that is present at the specified index. A refresh period is the maximum length of time that a user can use rights before the rights are refreshed from the server.

16.2.2.2 Syntax
```
getIRMRefreshPeriod(pindex)
```

16.2.2.3 Examples
The following example displays the refresh period that is present at index zero:

```
wls:/base_domain/serverConfig> getIRMRefreshPeriod(0)
```

The following example displays the refresh period that is present at index one:

```
wls:/base_domain/serverConfig> getIRMRefreshPeriod(1)
```

16.2.3 getIRMRefreshPeriods
Online command that displays all existing refresh periods.

16.2.3.1 Description
This command displays all existing refresh periods. A refresh period is the maximum length of time that a user can use rights before the rights are refreshed from the server.

16.2.3.2 Syntax
```
getIRMRefreshPeriods()
```

16.2.3.3 Example
```
wls:/base_domain/serverConfig> getIRMRefreshPeriods()
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>duration</td>
<td>Specifies the value of the refresh period. Integer.</td>
</tr>
<tr>
<td>dtype</td>
<td>Specifies the unit of the refresh period. 'MINUTES', 'HOURS', 'DAYS', 'MONTHS', 'YEARS'.</td>
</tr>
</tbody>
</table>
16.2.4 removeIRMRefreshPeriod

Online command that removes an existing refresh period.

16.2.4.1 Description
This command removes a refresh period that is present at the specified index. A refresh period is the maximum length of time that a user can use rights before the rights are refreshed from the server.

16.2.4.2 Syntax
removeIRMRefreshPeriod(pindex)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>pindex</td>
<td>Specifies the index of the refresh period.</td>
</tr>
</tbody>
</table>

16.2.4.3 Examples
The following example removes the refresh period that is present at index zero:

```
wls:/base_domain/serverConfig> removeIRMRefreshPeriod(0)
```

The following example removes the refresh period that is present at index one:

```
wls:/base_domain/serverConfig> removeIRMRefreshPeriod(1)
```

16.2.5 updateIRMRefreshPeriod

Online command that updates an existing refresh period.

16.2.5.1 Description
This command updates an existing refresh period. A refresh period is the maximum length of time that a user can use rights before the rights are refreshed from the server.

16.2.5.2 Syntax
updateIRMRefreshPeriod(pindex, duration, dtype)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>pindex</td>
<td>Specifies the index of the refresh period.</td>
</tr>
<tr>
<td>duration</td>
<td>Specifies the value of the refresh period. Integer.</td>
</tr>
<tr>
<td>dtype</td>
<td>Specifies the unit of the refresh period. 'MINUTES', 'HOURS', 'DAYS', 'MONTHS' or 'YEARS'.</td>
</tr>
</tbody>
</table>

16.2.5.3 Examples
The following example updates the refresh period at index zero to have a duration of 5 hours:

```
wls:/base_domain/serverConfig> updateIRMRefreshPeriod(0,5,"HOURS")
```

The following example updates the refresh period at index zero to have a duration of 50 minutes:

```
wls:/base_domain/serverConfig> updateIRMRefreshPeriod(0,50,"MINUTES")
```
16.2.6  addIRMSyncWindow

Online command that creates a sync window.

16.2.6.1 Description
This command creates a sync window. A sync window is a period during which Oracle IRM Desktop will attempt to contact the server to synchronize rights.

16.2.6.2 Syntax
addIRMSyncWindow(day, stHrs, stMins, endHrs, endMins)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>day</td>
<td>Specifies the weekday. 'MONDAY', 'TUESDAY', etc.</td>
</tr>
<tr>
<td>stHrs</td>
<td>Specifies the start hours. Integer.</td>
</tr>
<tr>
<td>stMins</td>
<td>Specifies the start minutes. Integer.</td>
</tr>
<tr>
<td>endHrs</td>
<td>Specifies the end hours. Integer.</td>
</tr>
<tr>
<td>endMins</td>
<td>Specifies the end minutes. Integer.</td>
</tr>
</tbody>
</table>

16.2.6.3 Example
The following example creates a sync window that will result in Oracle IRM Desktop attempting to contact the server between 9.30am and 6.30pm on Mondays:

wls:/base_domain/serverConfig> addIRMSyncWindow("MONDAY", 9, 30, 6, 30)

16.2.7  getIRMSyncWindow

Online command that displays an existing sync window.

16.2.7.1 Description
This command displays a sync window that is present at the specified index. A sync window is a period during which Oracle IRM Desktop will attempt to contact the server to synchronize rights.

16.2.7.2 Syntax
getIRMSyncWindow(sindex)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>sindex</td>
<td>Specifies the index of the sync window.</td>
</tr>
</tbody>
</table>

16.2.7.3 Examples
The following example displays the sync window at index zero:

wls:/base_domain/serverConfig> getIRMSyncWindow(0)

The following example displays the sync window at index one:

wls:/base_domain/serverConfig> getIRMSyncWindow(1)

16.2.8  getIRMSyncWindows

Online command that displays all existing sync windows.
16.2.8.1 Description
This command displays all existing sync windows. A sync window is a period during which Oracle IRM Desktop will attempt to contact the server to synchronize rights.

16.2.8.2 Syntax
getIRMSyncWindows()

16.2.8.3 Example
wls:/base_domain/serverConfig> getIRMSyncWindows()

16.2.9 removeIRMSyncWindow
Online command that removes an existing sync window.

16.2.9.1 Description
This command removes a sync window that is present at the specified index. A sync window is a period during which Oracle IRM Desktop will attempt to contact the server to synchronize rights.

16.2.9.2 Syntax
removeIRMSyncWindow(indexOfDay)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>sindex</td>
<td>Specifies the index of the sync window.</td>
</tr>
</tbody>
</table>

16.2.9.3 Examples
The following example removes the sync window at index zero:
wls:/base_domain/serverConfig> removeIRMSyncWindow(0)

The following example removes the sync window at index one:
wls:/base_domain/serverConfig> removeIRMSyncWindow(1)

16.2.10 updateIRMSyncWindow
Online command that updates an existing sync window.

16.2.10.1 Description
This command updates an existing sync window. A sync window is a period during which Oracle IRM Desktop will attempt to contact the server to synchronize rights.

16.2.10.2 Syntax
updateIRMSyncWindow(indexOfDay, day, stHrs, stMins, endHrs, endMins)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>indexOfDay</td>
<td>Specifies the index of the sync window. Integer.</td>
</tr>
<tr>
<td>day</td>
<td>Specifies the weekday. 'MONDAY', 'TUESDAY', etc.</td>
</tr>
<tr>
<td>stHrs</td>
<td>Specifies the start hours. Integer.</td>
</tr>
<tr>
<td>stMins</td>
<td>Specifies the start minutes. Integer.</td>
</tr>
</tbody>
</table>
16.2.10.3 Example
The following example updates the sync window at index zero so that Oracle IRM Desktop will attempt to contact the server between 9.30am and 5.30pm on Mondays:

```
wlsl/base_domain/serverConfig> updateIRMSyncWindow(0, "MONDAY", 9, 30, 5, 30)
```

16.2.11 getIRMCryptoSchema
Online command that displays the cryptography algorithm.

16.2.11.1 Description
This command displays the cryptography algorithm currently applied to files that are sealed using Oracle IRM.

16.2.11.2 Syntax
```
getIRMCryptoSchema()
```

16.2.11.3 Example
```
wlsl/base_domain/serverConfig> getIRMCryptoSchema()
```

16.2.12 setIRMCryptoSchema
Online command that sets the cryptography algorithm.

16.2.12.1 Description
This command sets the cryptography algorithm that will be applied to files that are sealed using Oracle IRM. The default of AES128 is recommended.

16.2.12.2 Syntax
```
setIRMCryptoSchema(cryptID)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>cryptID</td>
<td>Specifies the name of the cryptography algorithm. Possible algorithm names are AES128, AES256, AES128-FIPS, AES256-FIPS, DES3-FIPS.</td>
</tr>
</tbody>
</table>

16.2.12.3 Example
The following example sets the cryptography algorithm used for Oracle IRM communications to AES128:

```
wlsl/base_domain/serverConfig> setIRMCryptoSchema("AES128")
```

16.2.13 getIRMDeviceCount
Online command that displays the device count.
16.2.13.1 Description
This command displays the maximum number of devices on which a user can open a sealed document at one time. The value applies to all users, and does not differ for individual users.

16.2.13.2 Syntax
getIRMDeviceCount()

16.2.13.3 Example
wls:/base_domain/serverConfig> getIRMDeviceCount()

16.2.14 setIRMDeviceCount
Online command that sets the device count.

16.2.14.1 Description
This command sets the maximum number of devices on which a user can open a sealed document at one time. The value applies to all users. The device count is normally kept low (1 or 2) to make it difficult to circumvent document access restrictions by sharing passwords.

16.2.14.2 Syntax
setIRMDeviceCount(devCount)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>devCount</td>
<td>Specifies the device count value (the number of devices). Integer.</td>
</tr>
</tbody>
</table>

16.2.14.3 Example
The following example sets the device count to 2:

wls:/base_domain/serverConfig> setIRMDeviceCount(2)

16.2.15 getIRMJournalCleanUp
Online command that displays the current report record clean-up values.

16.2.15.1 Description
This command displays the report record clean-up values. The values show how often report record clean-ups are performed, and the maximum age of the report records before they are deleted.

16.2.15.2 Syntax
getIRMJournalCleanUp()

16.2.15.3 Example
The following example displays the current report record clean-up values:

wls:/base_domain/serverConfig> getIRMJournalCleanUp()

16.2.16 setIRMJournalCleanUp
Online command that sets report record clean-up values.
16.2.16.1 Description
This command sets how often report record clean-ups are performed, and the maximum age of report records before they are deleted.

16.2.16.2 Syntax
```
setIRMJournalCleanUp(clDuration, clUnitType, retDuration, retUnitType)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>clDuration</td>
<td>Specifies the value for how often report record clean-ups are performed. Integer.</td>
</tr>
<tr>
<td>clUnitType</td>
<td>Specifies the unit for how often report record clean-ups are performed. 'MINUTES', 'HOURS', 'DAYS', 'MONTHS', 'YEARS'.</td>
</tr>
<tr>
<td>retDuration</td>
<td>Specifies the value for the maximum age of report records before they are deleted. Integer.</td>
</tr>
<tr>
<td>retUnitType</td>
<td>Specifies the unit for the maximum age of report records before they are deleted. 'MINUTES', 'HOURS', 'DAYS', 'MONTHS', 'YEARS'.</td>
</tr>
</tbody>
</table>

16.2.16.3 Example
The following example runs report record clean-ups every 5 days, and deletes report records that are 6 months old:
```
wls:/base_domain/serverConfig> setIRMJournalCleanUp(5, "DAYS", 6, "MONTHS")
```

16.2.17 getIRMLicenseStateCleanUp
Online command that displays the license state clean-up frequency.

16.2.17.1 Description
This command displays the license state clean-up frequency (the frequency at which license records will be deleted).

16.2.17.2 Syntax
```
getIRMLicenseStateCleanUp()
```

16.2.17.3 Example
```
wls:/base_domain/serverConfig> getIRMLicenseStateCleanUp()
```

16.2.18 setIRMLicenseStateCleanUp
Online command that sets the license state clean-up frequency.

16.2.18.1 Description
This command sets the license state clean-up frequency (the frequency at which license records will be deleted).

16.2.18.2 Syntax
```
setIRMLicenseStateCleanUp(duration, unitType)
```
### 16.2.18.3 Examples

The following example sets the frequency at which license records will be deleted to 10 hours:

```
   wls://base_domain/serverConfig> setIRMLicenseStateCleanUp(10, "HOURS")
```

The following example sets the frequency at which license records will be deleted to 50 minutes:

```
   wls://base_domain/serverConfig> setIRMLicenseStateCleanUp(50, "MINUTES")
```

### 16.2.19 getIRMPrivacyURL

Online command that displays the URL of the privacy statement page.

#### 16.2.19.1 Description

This command displays the URL of the privacy statement page. The privacy statement page displays a statement that users must accept before viewing sealed content.

#### 16.2.19.2 Syntax

```
   getIRMPrivacyURL()
```

#### 16.2.19.3 Examples

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

### 16.2.20 setIRMPrivacyURL

Online command that sets the URL of the privacy statement page.

#### 16.2.20.1 Description

This command sets the URL of a privacy statement that users must accept before viewing sealed content.

#### 16.2.20.2 Syntax

```
   setIRMPrivacyURL(privacyURL)
```

#### Argument Definition

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>privacyURL</td>
<td>Specifies the URL of the privacy statement page.</td>
</tr>
</tbody>
</table>

#### 16.2.20.3 Example

The following example sets the URL of the privacy policy page to "http://irm.example.com/":

```
   wls://base_domain/serverConfig> setIRMPrivacyURL("http://irm.example.com")
```
16.2.21 getIRMKeyStore

Online command that displays the type and location of the Oracle IRM keystore.

16.2.21.1 Description
This command displays the type and location of the Oracle IRM keystore.

16.2.21.2 Syntax
getIRMKeyStore()

16.2.21.3 Examples
wls:/base_domain/serverConfig> getIRMKeyStore()

16.2.22 setIRMKeyStore

Online command that sets the type and location of the Oracle IRM keystore.

16.2.22.1 Description
This command sets the type and location of the Oracle IRM keystore. You should not normally need to change the keystore type and location from the default (type JKS at location ${domain.home}/config/fmwconfig/irm.jks).

16.2.22.2 Syntax
setIRMKeyStore()

You will be prompted to provide the following arguments:

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>KeyStore Type</td>
<td>Specifies the type of the keystore.</td>
</tr>
<tr>
<td>KeyStore Location</td>
<td>Specifies the location of the keystore.</td>
</tr>
</tbody>
</table>

16.2.22.3 Example
The following example sets the keystore type to JCEKS and the keystore location to D:/exampledir/:

wls:/base_domain/serverConfig> setIRMKeyStore()
Enter KeyStore Type: JCEKS
Enter KeyStore Location: D:/exampledir/

16.3 Migration Commands

Use the WLST commands listed in Table 16–3 to migrate user data between instances of Oracle IRM Server.

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>preIRMUserStoreUpgrade</td>
<td>Create a backup of Oracle IRM user data.</td>
<td>Online</td>
</tr>
<tr>
<td>postIRMUserStoreUpgrade</td>
<td>Migrate Oracle IRM user data.</td>
<td>Online</td>
</tr>
<tr>
<td>transferIRMAccount</td>
<td>Transfer an Oracle IRM account.</td>
<td>Online</td>
</tr>
</tbody>
</table>
16.3.1 **preIRMUserStoreUpgrade**

Online command that creates a backup of Oracle IRM user data.

16.3.1.1 **Description**

This command creates a backup of users and groups being used by Oracle IRM. It generates a data file that stores the details of users and groups.

16.3.1.2 **Syntax**

`preIRMUserStoreUpgrade()`

You will be prompted to provide the following arguments:

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server URL</td>
<td>Specifies the URL of the server, including the port number.</td>
</tr>
<tr>
<td>Username</td>
<td>Specifies the username to connect to the Oracle IRM instance.</td>
</tr>
<tr>
<td>Password</td>
<td>Specifies the password to connect to the Oracle IRM instance.</td>
</tr>
</tbody>
</table>

16.3.1.3 **Example**

```
wls:/base_domain/serverConfig> preIRMUserStoreUpgrade()
Server URL: t3://irm.example.com:1234
Username: irmuser
Password: irmuser_pwd
```

16.3.2 **postIRMUserStoreUpgrade**

Online command that migrates Oracle IRM user data.

16.3.2.1 **Description**

This command migrates Oracle IRM users and groups to be used with a new installation. It uses the data file generated by the `preIRMUserStoreUpgrade` command.

16.3.2.2 **Syntax**

`postIRMUserStoreUpgrade()`

You will be prompted to provide the following arguments:

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server URL</td>
<td>Specifies the URL of the server, including the port number.</td>
</tr>
<tr>
<td>Username</td>
<td>Specifies the username to connect to the Oracle IRM instance.</td>
</tr>
<tr>
<td>Password</td>
<td>Specifies the password to connect to the Oracle IRM instance.</td>
</tr>
</tbody>
</table>

16.3.2.3 **Example**

```
wls:/base_domain/serverConfig> postIRMUserStoreUpgrade()
Server URL: t3://irm.example.com:1234
Username: irmuser
Password: irmuser_pwd
```
16.3.3 transferIRMAccount

Online command that transfers an Oracle IRM account.

16.3.3.1 Description
This command transfers an existing Oracle IRM account into the specified account. It uses the data file generated by the preIRMUserStoreUpgrade command.

16.3.3.2 Syntax
transferIRMAccount(accName, accType, newAccName, newAccType)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>accName</td>
<td>Specifies the old account name.</td>
</tr>
<tr>
<td>accType</td>
<td>Specifies the old type of account (USER or GROUP).</td>
</tr>
<tr>
<td>newAccName</td>
<td>Specifies the new account name.</td>
</tr>
<tr>
<td>newAccType</td>
<td>Specifies the new type of account (USER or GROUP).</td>
</tr>
</tbody>
</table>

You will be prompted to supply the following arguments:

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server URL</td>
<td>Specifies the URL of the server, including the port number.</td>
</tr>
<tr>
<td>Username</td>
<td>Specifies the username to connect to the Oracle IRM instance.</td>
</tr>
<tr>
<td>Password</td>
<td>Specifies the password to connect to the Oracle IRM instance.</td>
</tr>
</tbody>
</table>

16.3.3.3 Example

\[ \text{wls:/base_domain/serverConfig>}
\text{transferIRMAccount("irmuser","USER","irmuser1","USER")}
\]

Server URL: t3://irm.example.com:1234
Username: irmuser
Password: irmuser_pwd

16.4 Test Content Commands

Use the WLST commands listed in Table 16–4 to set up test content for users of Oracle IRM Desktop.

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>addIRMTestContent</td>
<td>Create a new test content instance.</td>
<td>Online</td>
</tr>
<tr>
<td>getIRMTestContent</td>
<td>Display details for an existing test content instance.</td>
<td>Online</td>
</tr>
<tr>
<td>getIRMTestContents</td>
<td>Display details of all existing test content instances.</td>
<td>Online</td>
</tr>
<tr>
<td>removeIRMTestContent</td>
<td>Remove an existing test content instance.</td>
<td>Online</td>
</tr>
<tr>
<td>updateIRMTestContent</td>
<td>Update an existing test content instance.</td>
<td>Online</td>
</tr>
</tbody>
</table>

16.4.1 addIRMTestContent

Online command that creates a new test content instance.
16.4.1.1 Description
This command creates a test content instance. Test content instances identify an item of test content, usually an image file. Test content is shown in a sealed document when Oracle IRM Desktop successfully connects to Oracle IRM Server through the client test facility.

16.4.1.2 Syntax
addIRMTestContent(uri,localeKeys,testNames)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>uri</td>
<td>Specifies the URI of the test content (for example, an image file).</td>
</tr>
<tr>
<td>localeKeys</td>
<td>Specifies the locale(s) associated with this test content instance. Must be from the list of two-letter language codes given in Table 16–5 (for example, 'fr' for French). If there is more than one supported locale for an instance, the two-letter codes must be listed as comma-separated values.</td>
</tr>
<tr>
<td>testNames</td>
<td>Specifies the name(s) associated with this test content instance. If there is more than one name for a URI, they must be specified as comma-separated values.</td>
</tr>
</tbody>
</table>

Table 16–5 Language codes (ISO 639-1 “two-letter codes”)

<table>
<thead>
<tr>
<th>Language: en</th>
<th>Greek: el</th>
<th>Romanian: ro</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arabic: ar</td>
<td>Hebrew: iw</td>
<td>Russian: ru</td>
</tr>
<tr>
<td>Brazilian Portuguese: pt-BR</td>
<td>Hungarian: hu</td>
<td>Simplified Chinese: zh-CN</td>
</tr>
<tr>
<td>Czech: cs</td>
<td>Italian: it</td>
<td>Slovak: sk</td>
</tr>
<tr>
<td>Danish: da</td>
<td>Japanese: ja</td>
<td>Spanish: es</td>
</tr>
<tr>
<td>Dutch: nl</td>
<td>Korean: ko</td>
<td>Swedish: sv</td>
</tr>
<tr>
<td>English: en</td>
<td>Norwegian: no</td>
<td>Thai: th</td>
</tr>
<tr>
<td>Finnish: fi</td>
<td>Polish: pl</td>
<td>Traditional Chinese: zh-TW</td>
</tr>
<tr>
<td>German: de</td>
<td>Portuguese: pt</td>
<td>Turkish: tr</td>
</tr>
</tbody>
</table>

16.4.1.3 Examples
The following example creates a test content instance comprising an image named exampleImage.jpg at http://irm.example.com, for use with English installations, and showing the name 'Test Content':

wls:/base_domain/serverConfig>
addIRMTestContent("http://irm.example.com/exampleImage.jpg","en","Test Content")

The following example creates a test content instance comprising an image named exampleImage.jpg at http://irm.example.com, for use with English and French installations, and showing the names 'Test Content (en)' and 'Test Content (fr)'

wls:/base_domain/serverConfig>
addIRMTestContent("http://irm.example.com/exampleImage.jpg","en,fr","Test Content (en),Test Content (fr)")
16.4.2 **getIRMTestContent**

Online command that displays the details of an existing test content instance.

**16.4.2.1 Description**
This command displays the details of the test content instance that is present at the specified index. Test content instances identify an item of test content, usually an image file. Test content is shown in a sealed document when Oracle IRM Desktop successfully connects to Oracle IRM Server through the client test facility.

**16.4.2.2 Syntax**

```
getIRMTestContent(tindex)
```

**Argument Definition**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>tindex</td>
<td>Specifies the index of the test content instance.</td>
</tr>
</tbody>
</table>

**16.4.2.3 Examples**

The following example displays the details of the test content instance at index zero:

```
wls:/base_domain/serverConfig> getIRMTestContent(0)
```

The following example displays the details of the test content instance at index one:

```
wls:/base_domain/serverConfig> getIRMTestContent(1)
```

16.4.3 **getIRMTestContents**

Online command that displays all the test content instances.

**16.4.3.1 Description**
This command displays all the test content instances. Test content instances identify an item of test content, usually an image file. Test content is shown in a sealed document when Oracle IRM Desktop successfully connects to Oracle IRM Server through the client test page.

**16.4.3.2 Syntax**

```
getIRMTestContents()
```

**16.4.3.3 Example**

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

16.4.4 **removeIRMTestContent**

Online command that removes an existing test content instance.

**16.4.4.1 Description**
This command removes the test content instance that is present at the specified index. Test content instances identify an item of test content, usually an image file.

**16.4.4.2 Syntax**

```
removeIRMTestContent(tindex)
```
16.4.4.3 Examples
The following example removes the test content instance at index zero:

```
removeIRMTestContent(0)
```

The following example removes the test content instance at index one:

```
removeIRMTestContent(1)
```

16.4.5 updateIRMTestContent
Online command that updates an existing test content instance.

16.4.5.1 Description
This command updates an existing test content instance. Test content instances identify an item of test content, usually an image file. Test content is shown in a sealed document when Oracle IRM Desktop successfully connects to Oracle IRM Server through the client test facility.

16.4.5.2 Syntax
```
updateIRMTestContent(tindex, uri, localeKeys, testNames)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>tindex</td>
<td>Specifies the index of the test content instance. Integer.</td>
</tr>
<tr>
<td>uri</td>
<td>Specifies the URI of the test content (for example, an image file).</td>
</tr>
<tr>
<td>localeKeys</td>
<td>Specifies the locale(s) associated with this test content instance. Must be from the list of two-letter language codes given in Table 16–5 (for example, 'fr' for French). If there is more than one supported locale for an instance, the two-letter codes must be listed as comma-separated values.</td>
</tr>
<tr>
<td>testNames</td>
<td>Specifies the name(s) associated with this test content instance. If there is more than one name for a URI, they must be specified as comma-separated values.</td>
</tr>
</tbody>
</table>

16.4.5.3 Examples
The following example updates a test content instance by changing the image to exampleImage.jpg at http://irm.example.com, for use with English installations, and showing the name 'Test Content':

```
updateIRMTestContent(0, "http://irm.example.com/exampleImage.jpg", "en", "Test Content")
```

The following example updates a test content instance by changing the image to exampleImage.jpg at http://irm.example.com, for use with English and French installations, and showing the names 'Test Content (English)' and 'Test Content (French)'

```
updateIRMTestContent(0, "http://irm.example.com/exampleImage.jpg", "en,fr", "Test Content (English),Test Content (French)"
```
16.5 Languages Support Commands

Use the WLST commands listed in Table 16–6 to set up languages support for users of Oracle IRM Server.

Table 16–6  WLST Commands for Oracle IRM Server languages support

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>addIRMTranslation</td>
<td>Create a new language support instance.</td>
<td>Online</td>
</tr>
<tr>
<td>getIRMDefaultTranslation</td>
<td>Display the default language.</td>
<td>Online</td>
</tr>
<tr>
<td>getIRMTranslations</td>
<td>Display all the language support instances.</td>
<td>Online</td>
</tr>
<tr>
<td>removeIRMTranslation</td>
<td>Remove an existing language support instance.</td>
<td>Online</td>
</tr>
<tr>
<td>setIRMTranslations</td>
<td>Set the default language, and set a language support instance for one or more additional languages.</td>
<td>Online</td>
</tr>
</tbody>
</table>

16.5.1 addIRMTranslation

Online command that creates a new language support instance.

16.5.1.1 Description

This command creates a new language support instance. Each language support instance provides the facility in Oracle IRM Server to add names and descriptions in one or more languages (in addition to the default language).

16.5.1.2 Syntax

addIRMTranslation(transList)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>transList</td>
<td>Specifies the supported language(s). Must be from the list of two-letter language codes given in Table 16–5 (for example, 'fr' for French). If there is more than one supported language for an instance, the two-letter codes must be listed as comma-separated values.</td>
</tr>
</tbody>
</table>

16.5.1.3 Examples

The following example creates a language support instance that will enable users of Oracle IRM Server to add names and descriptions in French (in addition to their default language):

```
> wls:/base_domain/serverConfig> addIRMTranslation("fr")
```

The following example creates a language support instance that will enable users of Oracle IRM Server to add names and descriptions in French and Arabic (in addition to their default language):

```
> wls:/base_domain/serverConfig> addIRMTranslation("fr,ar")
```

16.5.2 getIRMDefaultTranslation

Online command that displays the default language.
16.5.2.1 Description
This command displays the default language.

16.5.2.2 Syntax
getIRMDefaultTranslation()

16.5.2.3 Example
wls:/base_domain/serverConfig> getIRMDefaultTranslation()

16.5.3 getIRMTranslations
Online command that displays all the language support instances.

16.5.3.1 Description
This command displays all the language support instances. Each language support instance provides the facility in Oracle IRM Server to add names and descriptions in one or more languages (in addition to the default language).

16.5.3.2 Syntax
getIRMTranslations()

16.5.3.3 Example
wls:/base_domain/serverConfig> getIRMTranslations()

16.5.4 removeIRMTranslation
Online command that removes an existing language support instance.

16.5.4.1 Description
This command removes the language support instance that is present at the specified index. Each language support instance provides the facility in Oracle IRM Server to add names and descriptions in one or more languages (in addition to the default language).

16.5.4.2 Syntax
removeIRMTranslation(tindex)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>tindex</td>
<td>Specifies the index of the language support instance.</td>
</tr>
</tbody>
</table>

16.5.4.3 Examples
The following example removes the language support instance at index zero:

wls:/base_domain/serverConfig> removeIRMTranslation(0)

The following example removes the language support instance at index one:

wls:/base_domain/serverConfig> removeIRMTranslation(1)
16.5.5 setIRMTranslations

Online command that sets the default language, and sets a language support instance for one or more languages in addition to the default language.

16.5.5.1 Description

This command sets the default language, and sets a language support instance for one or more languages in addition to the default language. Each language support instance provides the facility in Oracle IRM Server to add names and descriptions in one or more languages (in addition to the default language).

16.5.5.2 Syntax

setIRMTranslations(defaultTrans,transList)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>defaultTrans</td>
<td>Specifies the default language. Language code (for example, 'en' for English).</td>
</tr>
<tr>
<td>transList</td>
<td>Specifies the supported language(s). Must be from the list of two-letter language codes given in Table 16-5 (for example, 'fr' for French). If there is more than one supported language for an instance, the two-letter codes must be listed as comma-separated values.</td>
</tr>
</tbody>
</table>

16.5.5.3 Examples

The following example enables users of Oracle IRM Server to enter names and descriptions in English as the default language, and additionally to enter names and descriptions in French:

wls:/base_domain/serverConfig> setIRMTranslations("en","fr")

The following example enables users of Oracle IRM Server to enter names and descriptions in English as the default language, and additionally to enter names and descriptions in French and Arabic:

wls:/base_domain/serverConfig> setIRMTranslations("en","fr,ar")

16.6 Oracle IRM Desktop Installers Commands

Use the WLST commands listed in Table 16–7 to set up installation support for Oracle IRM Desktop software.

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>addIRMDownload</td>
<td>Create a new installer.</td>
<td>Online</td>
</tr>
<tr>
<td>getIRMDownload</td>
<td>Display the details for an existing installer.</td>
<td>Online</td>
</tr>
<tr>
<td>getIRMDownloads</td>
<td>Display the details for all installers.</td>
<td>Online</td>
</tr>
<tr>
<td>removeIRMDownload</td>
<td>Remove an existing installer.</td>
<td>Online</td>
</tr>
<tr>
<td>updateIRMDownload</td>
<td>Update an existing installer.</td>
<td>Online</td>
</tr>
</tbody>
</table>

16.6.1 addIRMDownload

Online command that creates a new installer.
16.6.1.1 Description
This command creates a new installer. Each installer identifies the locale and URI of software for installing Oracle IRM Desktop, and displays a name and version number that enables users of Oracle IRM Server to select the installer.

16.6.1.2 Syntax
```
addIRMDownload(locale,name,version,uri)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>locale</td>
<td>Specifies the locale of the installer. Must be from the list of two-letter language codes given in Table 16–5 (for example, 'en' for English).</td>
</tr>
<tr>
<td>name</td>
<td>Specifies the name for the installer.</td>
</tr>
<tr>
<td>version</td>
<td>Specifies the version of the installer. This is a label for the installer, and is not verified against the associated installation software.</td>
</tr>
<tr>
<td>uri</td>
<td>Specifies the URI of Oracle IRM Desktop installation software.</td>
</tr>
</tbody>
</table>

16.6.1.3 Example
The following example creates an installer for English language installation software at http://irm.example.com/, with the name 'Oracle IRM Desktop' and the version number 11.1.1.0.0 visible to users of Oracle IRM Server when they select this installer:
```
wls:/base_domain/serverConfig> addIRMDownload("en","Oracle IRM Desktop","11.1.1.0.0","http://irm.example.com")
```

16.6.2 getIRMDownload
Online command that displays the details for an existing installer.

16.6.2.1 Description
This command displays the details for an installer that is present at the specified index. Each installer identifies the locale and URI of software for installing Oracle IRM Desktop, and displays a name and version number that enables users of Oracle IRM Server to select the installer.

16.6.2.2 Syntax
```
getIRMDownload(dindex)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>dindex</td>
<td>Specifies the index of the download.</td>
</tr>
</tbody>
</table>

16.6.2.3 Examples
The following example displays the details for the installer at index zero:
```
wls:/base_domain/serverConfig> getIRMDownload(0)
```

The following example displays the details for the installer at index one:
```
wls:/base_domain/serverConfig> getIRMDownload(1)
```

16.6.3 getIRMDownloads
Online command that displays the details of all installers.
16.6.3.1 Description
This command displays the details of all installers. Each installer identifies the locale and URI of software for installing Oracle IRM Desktop, and displays a name and version number that enables users of Oracle IRM Server to select the installer.

16.6.3.2 Syntax
getIRMDownloads()

16.6.3.3 Example
wls:/base_domain/serverConfig> getIRMDownloads()

16.6.4 removeIRMDownload
Online command that removes an existing installer.

16.6.4.1 Description
Removes the installer that is present at the specified index. Each installer identifies the locale and URI of software for installing Oracle IRM Desktop, and displays a name and version number that enables users of Oracle IRM Server to select the installer.

16.6.4.2 Syntax
removeIRMDownload(dindex)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>dindex</td>
<td>Specifies the index of the download.</td>
</tr>
</tbody>
</table>

16.6.4.3 Examples
The following example removes the installer at index zero:

wls:/base_domain/serverConfig> removeIRMDownload(0)

The following example removes the installer at index one:

wls:/base_domain/serverConfig> removeIRMDownload(1)

16.6.5 updateIRMDownload
Online command that updates an existing installer.

16.6.5.1 Description
This command updates an existing installer. Each installer identifies the locale and URI of software for installing Oracle IRM Desktop, and displays a name and version number that enables users of Oracle IRM Server to select the installer.

16.6.5.2 Syntax
updateIRMDownload(dindex,locale,name,version,uri)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>dindex</td>
<td>Specifies the index of the installer. Integer.</td>
</tr>
</tbody>
</table>
16.6.5.3 Example
The following example updates the installer for index zero. After the update, the
installation software is English language and is located at http://irm.example.com/.
The name ‘Oracle IRM Desktop (English)’ and the version number 11.1.1.0.0 will be
visible to users of Oracle IRM Server when they select this installer.

```
wls:/base_domain/serverConfig> updateIRMDownload(0,"en","Oracle IRM Desktop
(English)"*,"11.1.1.0.0","http://irm.example.com/"
```
The following sections describe the WLST commands that are specific to Oracle Imaging and Process Management. Topics include:

- **Section 17.1, "Overview of I/PM WLST Command Categories"**
- **Section 17.2, "BPEL Diagnostic Commands"**
- **Section 17.3, "I/PM Configuration Commands"**

### 17.1 Overview of I/PM WLST Command Categories

WLST commands specific to Imaging and Process Management are divided into the following categories.

<table>
<thead>
<tr>
<th>Command category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Section 17.2, &quot;BPEL Diagnostic Commands&quot;</td>
<td>Return BPEL agent processing failures.</td>
</tr>
<tr>
<td>Section 17.3, &quot;I/PM Configuration Commands&quot;</td>
<td>Configure settings specific to Imaging and Process Management.</td>
</tr>
</tbody>
</table>

### 17.2 BPEL Diagnostic Commands

Use the I/PM WLST BPEL diagnostic commands, listed in table Table 17–2, to list and organize processing failures during BPEL processes.

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Section 17.2.1, &quot;sumIPMBpelFaults&quot;</td>
<td>Count processing failures during BPEL agent processing, grouped by choice of date, application ID, or batch ID.</td>
<td>Online</td>
</tr>
<tr>
<td>Section 17.2.2, &quot;clearIPMBpelFaults&quot;</td>
<td>Clear processing failures that occurred during BPEL agent processing.</td>
<td>Online</td>
</tr>
<tr>
<td>Section 17.2.3, &quot;listIPMBpelFaults&quot;</td>
<td>Provide details of processing failures that occurred during BPEL agent processing.</td>
<td>Online</td>
</tr>
<tr>
<td>Section 17.2.4, &quot;repairIPMBpelFaults&quot;</td>
<td>Repair processing failures that occurred during BPEL agent processing.</td>
<td>Online</td>
</tr>
</tbody>
</table>
17.2.1 `sumIPMBpelFaults`

Command Category: BPEL Diagnostic Commands

Use with WLST: Online

17.2.1.1 Description
Provides a count of processing failures that have occurred during BPEL agent processing. The results are grouped by date, application ID, or batch ID.

17.2.1.2 Syntax

```
sumIPMBpelFaults(group)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>groupOption</td>
<td>Required. One of the following:</td>
</tr>
<tr>
<td></td>
<td>■ DATE: Returns fault counts grouped by date.</td>
</tr>
<tr>
<td></td>
<td>■ APPID: Returns fault counts grouped by application ID.</td>
</tr>
<tr>
<td></td>
<td>■ BATCHID: Returns fault counts grouped by batch ID.</td>
</tr>
</tbody>
</table>

17.2.1.3 Example
The following example returns all BPEL faults grouped first by date, then by applications ID, then again grouped by batch ID.

```
sumIPMBpelFaults(group="DATE")
sumIPMBpelFaults(group="APPID")
sumIPMBpelFaults(group="BATCHID")
```

17.2.2 `clearIPMBpelFaults`

Command Category: BPEL Diagnostic Commands

Use with WLST: Online

17.2.2.1 Description
Clear processing failures that have occurred during BPEL agent processing.

17.2.2.2 Syntax

```
clearIPMBpelFaults([startDate], [endDate], [appId], [batchId])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>startDate</td>
<td>Optional. The start of the date range for which error details should be repaired, in yyyy-MM-dd format.</td>
</tr>
<tr>
<td>endDate</td>
<td>Optional. The end of the date range for which error details should be repaired, in yyyy-MM-dd format.</td>
</tr>
<tr>
<td>appId</td>
<td>Optional. The application ID for which error details should be repaired, in yyyy-MM-dd format.</td>
</tr>
<tr>
<td>batchId</td>
<td>Optional. The batch ID for which error details should be repaired.</td>
</tr>
</tbody>
</table>

17.2.2.3 Example
The following example clears the faults within the specified parameters.
17.2.3 listIPMBpelFaults

Command Category: BPEL Diagnostic Commands
Use with WLST: Online

17.2.3.1 Description
List details on processing failures that have occurred during BPEL agent processing.

17.2.3.2 Syntax
listIPMBpelFaults([startDate], [endDate], [appId], [batchId])

17.2.3.3 Example
The following example lists the faults within the specified parameters.
listIPMBpelFaults(startDate="2009-06-01", endDate="2009-06-02")
listIPMBpelFaults(appId=3)
listIPMBpelFaults(batchId=15)
listIPMBpelFaults(startDate="2009-06-01", endDate="2009-06-02", appId=3)

17.2.4 repairIPMBpelFaults

Command Category: BPEL Diagnostic Commands
Use with WLST: Online

17.2.4.1 Description
Repair processing failures that have occurred during BPEL agent processing.

17.2.4.2 Syntax
repairIPMBpelFaults([startDate], [endDate], [appId], [batchId])
I/PM Configuration Commands

17.2.4.3 Example
The following example repairs the faults within the specified parameters.

```java
repairIPMSpelFaults(startDate="2009-06-01", endDate="2009-06-02")
repairIPMSpelFaults(appId=3)
repairIPMSpelFaults(batchId=15)
repairIPMSpelFaults(startDate="2009-06-01", endDate="2009-06-02", appId=3)
```

17.3 I/PM Configuration Commands

Use the I/PM configuration commands, listed in Table 17–3, to list and set configuration values specific to I/PM.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appId</td>
<td>Optional. The application ID for which error details should be repaired.</td>
</tr>
<tr>
<td>batchId</td>
<td>Optional. The batch ID for which error details should be repaired.</td>
</tr>
</tbody>
</table>

### Table 17–3 Configuration Commands for I/PM

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Section 17.3.1,</td>
<td>Get an I/PM configuration setting value, similar to navigating to the custom I/PM config</td>
<td>Online</td>
</tr>
<tr>
<td>&quot;getIPMConfig&quot;</td>
<td>mbean and using the standard WLST set command.</td>
<td></td>
</tr>
<tr>
<td>Section 17.3.2,</td>
<td>Grants CredentialAccessPermissions to I/PM when I/PM managed servers are in a separate</td>
<td>Online</td>
</tr>
<tr>
<td>&quot;grantIPMCredAccess&quot;</td>
<td>domain home from the admin server.</td>
<td></td>
</tr>
<tr>
<td>Section 17.3.3,</td>
<td>List I/PM configuration mbeans.</td>
<td>Online</td>
</tr>
<tr>
<td>&quot;listIPMConfig&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Section 17.3.4,</td>
<td>Refresh security items currently stored in the I/PM database.</td>
<td>Online</td>
</tr>
<tr>
<td>&quot;refreshIPMSecurity&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Section 17.3.5,</td>
<td>Set an I/PM configuration value.</td>
<td>Online</td>
</tr>
<tr>
<td>&quot;setIPMConfig&quot;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

17.3.1 getIPMConfig

Command Category: I/PM Configuration Commands

Use with WLST: Online

17.3.1.1 Description

Gets an I/PM configuration setting value. The command is equivalent to browsing the custom mbean hierarchy to the I/PM config mbean and using the standard WLST set command to set an mbean attribute.

17.3.1.2 Syntax

```java
getIPMConfig(attrName)
```

### Argument Definition

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>attrName</td>
<td>Required. Name of the attribute to be read. Must be enclosed in single or</td>
</tr>
<tr>
<td></td>
<td>double quotes.</td>
</tr>
</tbody>
</table>
17.3.1.3 Example
The following example returns the value for the specified attribute names.

getIPMConfig('AgentUser')
getIPMConfig('CheckInterval')

17.3.2 grantIPMCredAccess
Grants CredentialAccessPermissions to I/PM so that it can read credentials from the credential store. This command is required in configurations where I/PM managed servers are in a separate domain home from the admin server. When at least one I/PM managed server is in the same domain home as the admin server, this command is not required, as CredentialAccessPermissions are granted during I/PM startup.

When the I/PM managed server is not in the same domain home as the admin server, however, the I/PM startup grant only affects the local settings. Local settings get overwritten when the admin server synchronizes its copy as the domain wide configuration, so this command updates the admin server configuration such that permissions are distributed correctly to all domain nodes.

17.3.2.1 Syntax
grantIPMCredAccess()

17.3.2.2 Example
The following example returns a list of all I/PM configuration mbeans.

grantIPMCredAccess()

17.3.3 listIPMConfig
Command Category: I/PM Configuration Commands
Use with WLST: Online

17.3.3.1 Description
Provides a listing of I/PM configuration mbeans. The command is equivalent to browsing the custom mbean hierarchy and listing the I/PM mbean attributes.

17.3.3.2 Syntax
listIPMConfig()

17.3.3.3 Example
The following example returns a list of all I/PM configuration mbeans.

listIPMConfig()

17.3.4 refreshIPMSecurity
Command Category: I/PM Configuration Commands
Use with WLST: Online
17.3.4.1 Description
Refreshes security items currently stored in the I/PM database. This is typically done when migrating security to a different policy store and only updates security items found in the new policy store.

17.3.4.2 Syntax
refreshIPMSecurity()

17.3.4.3 Example
The following example refreshes the security items stored in the I/PM database.
refreshIPMSecurity()

17.3.5 setIPMConfig
Command Category: I/PM Configuration Commands
Use with WLST: Online

17.3.5.1 Description
Sets an I/PM configuration setting value. The command is equivalent to browsing the custom mbean hierarchy to the I/PM config mbean and using the standard WLST 'set' command to set an mbean attribute.

17.3.5.2 Syntax
setIPMConfig(attrName, value)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>attrName</td>
<td>Required. Name of the attribute to be set. Must be enclosed in single or double quotes.</td>
</tr>
<tr>
<td>value</td>
<td>Required. Value of the attribute to be set. Only enclosed in single or double quotes if value is a string literal.</td>
</tr>
</tbody>
</table>

17.3.5.3 Example
The following example sets the specified values for the specified attribute names.

setIPMConfig('AgentUser', 'agentadmin')
setIPMConfig('CheckInterval', 30)