

Oracle® Fusion Middleware

WLST Command Reference for SOA Suite



12c Release (12.2.1.3.0)

E96877-03

August 2019

The Oracle logo, consisting of a solid red square with the word "ORACLE" in white, uppercase, sans-serif font centered within it.

ORACLE®

Copyright © 2007, 2019, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

Preface

Audience	xix
Documentation Accessibility	xix
Related Documents	xix
Conventions	xx

1 Oracle SOA Suite Custom WLST Commands

1.1 Overview of WSLT Command Categories	1-1
1.2 Deployment Commands	1-2
1.2.1 sca_deployComposite	1-3
1.2.1.1 Description	1-3
1.2.1.2 Syntax	1-3
1.2.1.3 Examples	1-4
1.2.2 sca_undeployComposite	1-5
1.2.2.1 Description	1-5
1.2.2.2 Syntax	1-5
1.2.2.3 Examples	1-5
1.3 WLST MDS Commands	1-6
1.3.1 importMetadata	1-6
1.3.2 exportMetadata	1-7
1.4 SOA Composite Application Management Commands	1-9
1.4.1 sca_startComposite	1-9
1.4.1.1 Description	1-9
1.4.1.2 Syntax	1-9
1.4.1.3 Example	1-10
1.4.2 sca_stopComposite	1-10
1.4.2.1 Description	1-10
1.4.2.2 Syntax	1-10
1.4.2.3 Example	1-11
1.4.3 sca_activateComposite	1-11
1.4.3.1 Description	1-11
1.4.3.2 Syntax	1-11

1.4.3.3	Example	1-12
1.4.4	sca_retireComposite	1-12
1.4.4.1	Description	1-12
1.4.4.2	Syntax	1-12
1.4.4.3	Example	1-13
1.4.5	sca_assignDefaultComposite	1-13
1.4.5.1	Description	1-13
1.4.5.2	Syntax	1-13
1.4.5.3	Example	1-14
1.4.6	sca_listDeployedComposites	1-14
1.4.6.1	Description	1-14
1.4.6.2	Syntax	1-14
1.4.6.3	Example	1-14
1.5	Configuration Plan Management Commands	1-14
1.5.1	sca_attachPlan	1-15
1.5.1.1	Description	1-15
1.5.1.2	Syntax	1-15
1.5.1.3	Examples	1-15
1.5.2	sca_extractPlan	1-16
1.5.2.1	Description	1-16
1.5.2.2	Syntax	1-16
1.5.2.3	Example	1-16
1.5.3	sca_generatePlan	1-16
1.5.3.1	Description	1-17
1.5.3.2	Syntax	1-17
1.5.3.3	Examples	1-17
1.5.4	sca_validatePlan	1-17
1.5.4.1	Description	1-17
1.5.4.2	Syntax	1-18
1.5.4.3	Examples	1-18
1.6	Task Validation Commands	1-18
1.6.1	sca_validateTask	1-18
1.6.1.1	Description	1-19
1.6.1.2	Syntax	1-19
1.6.1.3	Example	1-19
1.7	SOA Composite Application Compilation Commands	1-19
1.7.1	sca_setProp	1-19
1.7.1.1	Description	1-19
1.7.1.2	Syntax	1-20
1.7.1.3	Example	1-20
1.7.2	sca_compile	1-20

1.7.2.1	Description	1-20
1.7.2.2	Syntax	1-20
1.7.2.3	Examples	1-21
1.8	SOA Composite Application Packaging Commands	1-21
1.8.1	sca_package	1-21
1.8.1.1	Description	1-21
1.8.1.2	Syntax	1-22
1.8.1.3	Examples	1-22
1.9	SOA Composite Application Test Commands	1-22
1.9.1	sca_test	1-23
1.9.1.1	Description	1-23
1.9.1.2	Syntax	1-23
1.9.1.3	Examples	1-23
1.10	SOA Composite Application HTTP Client-Based Export and Import Commands	1-23
1.10.1	sca_exportComposite	1-24
1.10.1.1	Description	1-24
1.10.1.2	Syntax	1-24
1.10.1.3	Examples	1-25
1.10.2	sca_exportUpdates	1-25
1.10.2.1	Description	1-25
1.10.2.2	Syntax	1-25
1.10.2.3	Examples	1-26
1.10.3	sca_importUpdates	1-27
1.10.3.1	Description	1-27
1.10.3.2	Syntax	1-27
1.10.3.3	Examples	1-27
1.10.4	sca_exportSharedData	1-27
1.10.4.1	Description	1-28
1.10.4.2	Syntax	1-28
1.10.4.3	Examples	1-28
1.10.5	sca_removeSharedData	1-28
1.10.5.1	Description	1-28
1.10.5.2	Syntax	1-28
1.10.5.3	Examples	1-29
1.11	SOA Composite Application MBean-Based Export and Import Commands	1-29
1.11.1	sca_exportCompositeMb	1-30
1.11.1.1	Description	1-30
1.11.1.2	Syntax	1-30
1.11.1.3	Examples	1-30
1.11.2	sca_exportUpdatesMb	1-31

1.11.2.1	Description	1-31
1.11.2.2	Syntax	1-31
1.11.2.3	Examples	1-31
1.11.3	sca_importUpdatesMb	1-31
1.11.3.1	Description	1-31
1.11.3.2	Syntax	1-32
1.11.3.3	Examples	1-32
1.11.4	sca_exportSharedDataMb	1-32
1.11.4.1	Description	1-32
1.11.4.2	Syntax	1-32
1.11.4.3	Examples	1-32
1.12	SOA Composite Application Partition Management Commands	1-33
1.12.1	sca_createPartition	1-33
1.12.1.1	Description	1-33
1.12.1.2	Syntax	1-33
1.12.1.3	Examples	1-34
1.12.2	sca_deletePartition	1-34
1.12.2.1	Description	1-34
1.12.2.2	Syntax	1-34
1.12.2.3	Examples	1-34
1.12.3	sca_startCompositesInPartition	1-34
1.12.3.1	Description	1-34
1.12.3.2	Syntax	1-34
1.12.3.3	Examples	1-34
1.12.4	sca_stopCompositesInPartition	1-35
1.12.4.1	Description	1-35
1.12.4.2	Syntax	1-35
1.12.4.3	Examples	1-35
1.12.5	sca_activateCompositesInPartition	1-35
1.12.5.1	Description	1-35
1.12.5.2	Syntax	1-35
1.12.5.3	Examples	1-35
1.12.6	sca_retireCompositesInPartition	1-35
1.12.6.1	Description	1-36
1.12.6.2	Syntax	1-36
1.12.6.3	Examples	1-36
1.12.7	sca_listPartitions	1-36
1.12.7.1	Description	1-36
1.12.7.2	Syntax	1-36
1.12.7.3	Examples	1-36
1.12.8	sca_listCompositesInPartition	1-36

1.12.8.1	Description	1-36
1.12.8.2	Syntax	1-36
1.12.8.3	Examples	1-37
1.12.9	sca_createPartitionWMG	1-37
1.12.9.1	Description	1-37
1.12.9.2	Syntax	1-37
1.12.9.3	Examples	1-37
1.13	SOA Composite Application Offline Deployment Management Commands	1-37
1.13.1	sca_registerCompositeOfflineDeployment	1-38
1.13.1.1	Description	1-38
1.13.1.2	Syntax	1-39
1.13.1.3	Examples	1-39
1.13.2	sca_unregisterCompositeOfflineDeployment	1-39
1.13.2.1	Description	1-39
1.13.2.2	Syntax	1-39
1.13.2.3	Examples	1-40
1.14	SOA Health Check Commands	1-40
1.14.1	executeHealthCheckAndGenerateReport	1-42
1.14.2	executeHealthCheckCategoryAndGenerateReport	1-42
1.14.3	executeHealthChecksAndGenerateReport	1-43
1.14.4	getUnsuccessfulHCRequestIds	1-43
1.14.5	getUnsuccessfulHCRequestResultReport	1-44
1.14.6	listHealthChecks	1-44
1.14.7	listHealthChecksInCategory	1-45
1.14.8	listHealthCheckCategories	1-46
1.14.9	deleteAllHCRequestResults	1-46
1.14.10	deleteHCRequestResults	1-47
1.14.11	getUnsuccessfulHCResultReport	1-47
1.14.12	getHCPParameter	1-48
1.14.13	setHCPParameter	1-48
1.14.14	resetHCPParameter	1-49
1.14.15	getHCPParameters	1-49
1.14.16	enableHCStartupRun	1-50
1.14.17	setHCStartupRunCategory	1-50
1.14.18	getHCStartupRun	1-51
1.14.19	createHCPeriodicRun	1-51
1.14.20	enableHCPeriodicRun	1-52
1.14.21	setHCPeriodicRunInterval	1-52
1.14.22	setHCPeriodicRunSchedule	1-53
1.14.23	setHCPeriodicRunCategory	1-53
1.14.24	getHCPeriodicRun	1-54

1.14.25	getHCPublicRuns	1-54
1.14.26	removeHCPublicRun	1-55
1.14.27	getDisabledHealthChecks	1-56
1.14.28	addDisabledHealthCheck	1-56
1.14.29	removeDisabledHealthCheck	1-56

2 Oracle Business Process Management Custom WLST Commands

2.1	BPMLifecycleAdmin Command Group	2-1
2.1.1	create_public_share	2-1
2.1.1.1	Description	2-1
2.1.1.2	Syntax	2-1
2.1.1.3	Examples	2-2
2.1.2	unlock_public_share	2-2
2.1.2.1	Description	2-2
2.1.2.2	Syntax	2-3
2.1.2.3	Example	2-3
2.1.3	export_public_share	2-3
2.1.3.1	Description	2-3
2.1.3.2	Syntax	2-3
2.1.3.3	Example	2-4
2.1.4	delete_public_share	2-4
2.1.4.1	Description	2-4
2.1.4.2	Syntax	2-4
2.1.4.3	Examples	2-5
2.1.5	publish_template	2-5
2.1.5.1	Description	2-5
2.1.5.2	Syntax	2-5
2.1.5.3	Example	2-6
2.1.6	export_template	2-6
2.1.6.1	Description	2-6
2.1.6.2	Syntax	2-6
2.1.6.3	Example	2-7
2.1.7	delete_template	2-7
2.1.7.1	Description	2-7
2.1.7.2	Syntax	2-7
2.1.7.3	Example	2-8

3 Oracle Enterprise Scheduler Custom WLST Commands

3.1	Native Invocation	3-1
-----	-------------------	-----

3.1.1	Enabling and Disabling Print Statements	3-2
3.1.2	Getting Help	3-2
3.2	Oracle Enterprise Scheduler Custom Native Commands	3-2
3.2.1	Manage (Add/Delete/Modify/Get) Configuration Parameters	3-3
3.2.1.1	Description	3-3
3.2.1.2	Syntax	3-3
3.2.1.3	Examples	3-3
3.2.1.4	Related Shell Script	3-4
3.2.2	Get Log and Output Content of a Request	3-4
3.2.2.1	Description	3-4
3.2.2.2	Syntax	3-4
3.2.2.3	Examples	3-4
3.2.2.4	Related Shell Script	3-5
3.2.3	Search and List Requests	3-5
3.2.3.1	Description	3-5
3.2.3.2	Syntax	3-5
3.2.3.3	Examples	3-6
3.2.3.4	Shell Script	3-6
3.2.4	Manage Requests	3-7
3.2.4.1	Description	3-7
3.2.4.2	Syntax	3-7
3.2.4.3	Examples	3-7
3.2.4.4	Related Shell Script	3-7
3.2.5	Manage Oracle Enterprise Scheduler Servers	3-8
3.2.5.1	Description	3-8
3.2.5.2	Syntax	3-8
3.2.5.3	Examples	3-8
3.2.5.4	Related Shell Script	3-8
3.2.6	Submit Job Requests to Oracle Enterprise Scheduler	3-8
3.2.6.1	Description	3-8
3.2.6.2	Syntax	3-9
3.2.6.3	Examples	3-10
3.2.6.4	Related Shell Script	3-11
3.2.7	Manage Oracle Enterprise Scheduler Job Definitions	3-11
3.2.7.1	Description	3-11
3.2.7.2	Syntax	3-12
3.2.7.3	Examples	3-13
3.2.7.4	Related Shell Script	3-14
3.2.8	Manage Oracle Enterprise Scheduler Schedule Definitions	3-14
3.2.8.1	Description	3-14
3.2.8.2	Syntax	3-15

3.2.8.3	Examples	3-16
3.2.8.4	Related Shell Script	3-17
3.3	Oracle Enterprise Scheduler Batch Delete Requests	3-17
3.3.1	The batchDeleteSchedulerRequest Command	3-18
3.3.1.1	Description	3-18
3.3.1.2	Syntax	3-18
3.3.1.3	Examples	3-20
3.3.1.4	Related Shell Script	3-20
3.4	Work Allocation Commands	3-21
3.4.1	addProcessorBinding	3-21
3.4.1.1	Description	3-21
3.4.1.2	Syntax	3-21
3.4.1.3	Examples	3-22
3.4.2	createWorkshift	3-22
3.4.2.1	Description	3-22
3.4.2.2	Syntax	3-22
3.4.2.3	Example	3-23
3.4.3	deleteWorkAssignment	3-23
3.4.3.1	Description	3-23
3.4.3.2	Syntax	3-23
3.4.3.3	Example	3-23
3.4.4	deleteWorkshift	3-23
3.4.4.1	Description	3-24
3.4.4.2	Syntax	3-24
3.4.4.3	Example	3-24
3.4.5	queryProcessorBindings	3-24
3.4.5.1	Description	3-24
3.4.5.2	Syntax	3-24
3.4.5.3	Example	3-24
3.4.6	queryWorkAssignments	3-24
3.4.6.1	Description	3-25
3.4.6.2	Syntax	3-25
3.4.6.3	Examples	3-25
3.4.7	queryWorkSchedules	3-25
3.4.7.1	Description	3-25
3.4.7.2	Syntax	3-25
3.4.7.3	Examples	3-26
3.4.8	queryWorkshifts	3-26
3.4.8.1	Description	3-26
3.4.8.2	Syntax	3-26
3.4.8.3	Examples	3-26

3.4.9	removeProcessorBinding	3-27
3.4.9.1	Description	3-27
3.4.9.2	Syntax	3-27
3.4.9.3	Example	3-27
3.4.10	updateWorkshift	3-27
3.4.10.1	Description	3-27
3.4.10.2	Syntax	3-27
3.4.10.3	Examples	3-28
3.5	Oracle Enterprise Scheduler Diagnostic Dumps	3-28
3.5.1	The Dump Commands	3-29
3.5.1.1	executeDump	3-30
3.5.1.2	listDumps	3-30
3.5.1.3	describeDump	3-30
3.5.2	Dump Examples	3-31
3.5.3	Creating an Oracle Enterprise Scheduler Incident	3-32
3.6	Oracle Enterprise Scheduler Convenience Scripts	3-32
3.6.1	essManageRuntimeConfig	3-33
3.6.1.1	Syntax	3-33
3.6.1.2	Options	3-33
3.6.1.3	Associated Files	3-34
3.6.1.4	Exit Values	3-34
3.6.1.5	Examples	3-34
3.6.1.6	Related Native Command	3-34
3.6.2	essGetOutputContent	3-34
3.6.2.1	Syntax	3-34
3.6.2.2	Options	3-35
3.6.2.3	Associated Files	3-35
3.6.2.4	Exit Values	3-35
3.6.2.5	Examples	3-35
3.6.2.6	Related Native Command	3-36
3.6.3	essQueryRequests	3-36
3.6.3.1	Syntax	3-36
3.6.3.2	Options	3-36
3.6.3.3	Associated Files	3-37
3.6.3.4	Exit Values	3-37
3.6.3.5	Examples	3-38
3.6.3.6	Related Native Command	3-38
3.6.4	essManageRequests	3-38
3.6.4.1	Syntax	3-38
3.6.4.2	Options	3-38
3.6.4.3	Associated Files	3-39

3.6.4.4	Exit Values	3-39
3.6.4.5	Examples	3-39
3.6.4.6	Related Native Command	3-40
3.6.5	essManageServer	3-40
3.6.5.1	Syntax	3-40
3.6.5.2	Options	3-40
3.6.5.3	Associated Files	3-40
3.6.5.4	Exit Values	3-41
3.6.5.5	Examples	3-41
3.6.5.6	Related Native Command	3-41
3.6.6	essSubmitRequest	3-41
3.6.6.1	Syntax	3-42
3.6.6.2	Options	3-42
3.6.6.3	Associated Files	3-43
3.6.6.4	Exit Values	3-43
3.6.6.5	Examples	3-43
3.6.6.6	Related Native Command	3-44
3.6.7	essManageJobDefn	3-44
3.6.7.1	Syntax	3-45
3.6.7.2	Options	3-45
3.6.7.3	Associated Files	3-46
3.6.7.4	Exit Values	3-46
3.6.7.5	Examples	3-47
3.6.7.6	Related Native Command	3-48
3.6.8	essManageSchedule	3-48
3.6.8.1	Syntax	3-48
3.6.8.2	Options	3-48
3.6.8.3	Associated Files	3-49
3.6.8.4	Exit Values	3-50
3.6.8.5	Examples	3-50
3.6.8.6	Related Native Command	3-51
3.6.9	essBatchDeleteRequests	3-51
3.6.9.1	Syntax	3-51
3.6.9.2	Options	3-51
3.6.9.3	Associated Files	3-52
3.6.9.4	Exit Values	3-52
3.6.9.5	Examples	3-52
3.6.9.6	Related Native Command	3-53

4 Oracle Managed File Transfer Custom WLST Commands

4.1	Overview of MFT WLST Command Categories	4-1
4.2	MFT Artifact Management Commands	4-2
4.2.1	bulkDeployArtifact	4-3
4.2.1.1	Description	4-3
4.2.1.2	Syntax	4-3
4.2.1.3	Example	4-3
4.2.2	createArtifacts	4-3
4.2.3	deleteArtifact	4-4
4.2.3.1	Description	4-4
4.2.3.2	Syntax	4-4
4.2.3.3	Example	4-4
4.2.4	deleteArtifactDeployment	4-4
4.2.4.1	Description	4-5
4.2.4.2	Syntax	4-5
4.2.4.3	Example	4-5
4.2.5	deployArtifact	4-5
4.2.5.1	Description	4-5
4.2.5.2	Syntax	4-5
4.2.5.3	Example	4-6
4.2.6	disableArtifact	4-6
4.2.6.1	Description	4-6
4.2.6.2	Syntax	4-6
4.2.6.3	Example	4-6
4.2.7	enableArtifact	4-6
4.2.7.1	Description	4-7
4.2.7.2	Syntax	4-7
4.2.7.3	Example	4-7
4.2.8	exportDeployedArtifact	4-7
4.2.8.1	Description	4-7
4.2.8.2	Syntax	4-7
4.2.8.3	Example	4-8
4.2.9	isArtifactInMDS	4-8
4.2.9.1	Description	4-8
4.2.9.2	Syntax	4-8
4.2.9.3	Example	4-9
4.2.10	undeployArtifact	4-9
4.2.10.1	Description	4-9
4.2.10.2	Syntax	4-9
4.2.10.3	Example	4-9

4.3	MFT Metadata Commands	4-9
4.3.1	exportMftMetadata	4-10
4.3.1.1	Description	4-10
4.3.1.2	Syntax	4-10
4.3.1.3	Example	4-10
4.3.2	exportTransferMetadata	4-11
4.3.2.1	Description	4-11
4.3.2.2	Syntax	4-11
4.3.2.3	Example	4-11
4.3.3	importMftMetadata	4-11
4.3.3.1	Description	4-11
4.3.3.2	Syntax	4-12
4.3.3.3	Example	4-12
4.3.4	resetMetadata	4-12
4.3.4.1	Description	4-12
4.3.4.2	Syntax	4-12
4.3.4.3	Example	4-12
4.4	MFT Key Management Commands	4-13
4.4.1	deleteCSFKey	4-13
4.4.1.1	Description	4-13
4.4.1.2	Syntax	4-13
4.4.1.3	Example	4-14
4.4.2	exportCSFKey	4-14
4.4.2.1	Description	4-14
4.4.2.2	Syntax	4-14
4.4.2.3	Example	4-14
4.4.3	generateKeys	4-14
4.4.3.1	Description	4-14
4.4.3.2	Syntax	4-15
4.4.3.3	Example	4-15
4.4.4	importCSFKey	4-15
4.4.4.1	Description	4-15
4.4.4.2	Syntax	4-15
4.4.4.3	Example	4-16
4.4.5	listCSFKeyAliases	4-16
4.4.5.1	Description	4-16
4.4.5.2	Syntax	4-16
4.4.5.3	Example	4-17
4.4.6	updateCSFKey	4-17
4.4.6.1	Description	4-17
4.4.6.2	Syntax	4-17

4.4.6.3	Example	4-17
4.5	MFT Deployment History Commands	4-17
4.5.1	getSourceDeploymentHistory	4-18
4.5.1.1	Description	4-18
4.5.1.2	Syntax	4-18
4.5.1.3	Example	4-18
4.5.2	getTargetDeploymentHistory	4-18
4.5.2.1	Description	4-18
4.5.2.2	Syntax	4-19
4.5.2.3	Example	4-19
4.5.3	getTransferDeploymentHistory	4-19
4.5.3.1	Description	4-19
4.5.3.2	Syntax	4-19
4.5.3.3	Example	4-19
4.6	MFT Transfer Management Commands	4-19
4.6.1	getTransferInfo	4-20
4.6.1.1	Description	4-20
4.6.1.2	Syntax	4-20
4.6.1.3	Example	4-20
4.6.2	pauseTransfer	4-20
4.6.2.1	Description	4-21
4.6.2.2	Syntax	4-21
4.6.2.3	Example	4-21
4.6.3	resubmit	4-21
4.6.3.1	Description	4-21
4.6.3.2	Syntax	4-21
4.6.3.3	Example	4-22
4.6.4	resumeTransfer	4-22
4.6.4.1	Description	4-22
4.6.4.2	Syntax	4-22
4.6.4.3	Example	4-22
4.7	MFT Embedded Server Commands	4-23
4.7.1	ConfigureHomeDir	4-23
4.7.1.1	Description	4-23
4.7.1.2	Syntax	4-23
4.7.1.3	Example	4-23
4.7.2	grantPermissionToDirectory	4-24
4.7.2.1	Description	4-24
4.7.2.2	Syntax	4-24
4.7.2.3	Example	4-24
4.7.3	listAllPermissions	4-24

4.7.3.1	Description	4-25
4.7.3.2	Syntax	4-25
4.7.3.3	Example	4-25
4.7.4	revokePermissionForDirectory	4-25
4.7.4.1	Description	4-25
4.7.4.2	Syntax	4-25
4.7.4.3	Example	4-26
4.7.5	startEmbeddedServer	4-26
4.7.5.1	Description	4-26
4.7.5.2	Syntax	4-26
4.7.5.3	Example	4-26
4.7.6	stopEmbeddedServer	4-26
4.7.6.1	Description	4-26
4.7.6.2	Syntax	4-27
4.7.6.3	Example	4-27
4.7.7	updatePorts	4-27
4.7.7.1	Description	4-27
4.7.7.2	Syntax	4-27
4.7.7.3	Example	4-27
4.8	MFT Callout Commands	4-28
4.8.1	createCallouts	4-28
4.8.1.1	Description	4-28
4.8.1.2	Syntax	4-28
4.8.1.3	Example	4-28
4.8.2	deleteCallout	4-28
4.8.2.1	Description	4-29
4.8.2.2	Syntax	4-29
4.8.2.3	Example	4-29
4.8.3	listCallouts	4-29
4.8.3.1	Description	4-29
4.8.3.2	Syntax	4-29
4.8.3.3	Example	4-29
4.8.4	updateCallouts	4-29
4.8.4.1	Description	4-30
4.8.4.2	Syntax	4-30
4.8.4.3	Example	4-30
4.9	MFT Event Notification Commands	4-30
4.9.1	addContactToNotification	4-31
4.9.1.1	Description	4-31
4.9.1.2	Syntax	4-31
4.9.1.3	Example	4-31

4.9.2	createContact	4-31
4.9.2.1	Description	4-31
4.9.2.2	Syntax	4-32
4.9.2.3	Example	4-32
4.9.3	deleteContact	4-32
4.9.3.1	Description	4-32
4.9.3.2	Syntax	4-32
4.9.3.3	Example	4-33
4.9.4	listContacts	4-33
4.9.4.1	Description	4-33
4.9.4.2	Syntax	4-33
4.9.4.3	Example	4-33
4.9.5	removeContactFromNotification	4-33
4.9.5.1	Description	4-33
4.9.5.2	Syntax	4-33
4.9.5.3	Example	4-34
4.9.6	updateEvent	4-34
4.9.6.1	Description	4-34
4.9.6.2	Syntax	4-34
4.9.6.3	Example	4-34
4.10	MFT Archive and Restore Commands	4-35
4.10.1	archiveInstanceData	4-35
4.10.1.1	Description	4-35
4.10.1.2	Syntax	4-35
4.10.1.3	Example	4-36
4.10.2	restoreInstanceData	4-37
4.10.2.1	Description	4-37
4.10.2.2	Syntax	4-37
4.10.2.3	Example	4-38
4.10.3	archivePayloads	4-38
4.10.3.1	Description	4-38
4.10.3.2	Syntax	4-38
4.10.3.3	MBean Properties	4-38
4.10.3.4	Example	4-39
4.10.4	restorePayloadsByName	4-39
4.10.4.1	Description	4-39
4.10.4.2	Syntax	4-39
4.10.4.3	MBean Properties	4-39
4.10.4.4	Example	4-40
4.10.5	restorePayloadsByPrefix	4-40
4.10.5.1	Description	4-40

4.10.5.2	Syntax	4-40
4.10.5.3	MBean Properties	4-40
4.10.5.4	Example	4-41
4.11	MFT Purge Commands	4-41
4.11.1	purgeInstanceData	4-41
4.11.1.1	Description	4-41
4.11.1.2	Syntax	4-41
4.11.1.3	Example	4-42
4.11.2	purgePayloads	4-43
4.11.2.1	Description	4-43
4.11.2.2	Syntax	4-43
4.11.2.3	MBean Properties	4-43
4.11.2.4	Example	4-44
4.12	Setting System MBean Properties for MFT WLST Commands	4-44

Preface

WLST Command Reference for SOA Suite describes all of the Oracle SOA Suite, Business Process Management, Enterprise Scheduler, and Managed File Transfer WLST commands that are available to use with the WebLogic Scripting Tool (WLST).

Audience

This guide is intended for Oracle SOA Suite administrators and operators who deploy SOA applications. It is assumed that you are familiar with Web technologies and the operating system and platform where WebLogic Server and Fusion Middleware products are installed.

Documentation Accessibility

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

Accessible Access to Oracle Support

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

Related Documents

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

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

- *WLST Command Reference for WebLogic Server* for information about the WLST commands for WebLogic Server.
- *WLST Command Reference for Infrastructure Components* for information about the WLST commands that are available for Oracle Fusion Middleware core components, including Java Required Files (JRF), Web services, Metadata services (MDS), Application Development Framework (ADF), Dynamic Monitoring Service (DMS), Logging, Diagnostic Framework, User Messaging Service (UMS), and Enterprise Scheduling Service (ESS).
- *WLST Command Reference for Infrastructure Security* for information about the WLST commands that are available for Oracle Fusion Middleware Infrastructure Security components, including Auditing, SSL, Oracle Identity Federation,

Directory Integration Platform, Oracle Access Management (OAM), Oracle Security Token Service, and Oracle Keystore Service.

- *WebCenter WLST Command Reference* for information about the WLST commands that are available for WebCenter components, including WebCenter Portal, WebCenter Content, WebCenter Information Rights Management (IRM), and WebCenter Imaging Process Management (IPM).
- Using Ant Tasks to Configure and Use a WebLogic Server Domain in *Developing Applications for Oracle WebLogic Server* for information about using WebLogic Ant tasks for starting and stopping WebLogic Server instances and configuring WebLogic domains.
- Deployment Tools in *Deploying Applications to Oracle WebLogic Server* for information about tools that WebLogic Server provides for deploying applications and stand-alone modules.
- *Oracle WebLogic Server Administration Console Online Help* for information about a Web-based graphical user interface for managing and monitoring WebLogic domains.
- *Creating WebLogic Domains Using the Configuration Wizard* for information about using a graphical user interface to create a WebLogic domain or extend an existing one.
- *Creating Templates and Domains Using the Pack and Unpack Commands* for information about commands that recreate existing WebLogic domains quickly and easily.
- *Developing Custom Management Utilities Using JMX for Oracle WebLogic Server* for information about using Java Management Extensions (JMX) APIs to monitor and modify WebLogic Server resources.
- *Monitoring Oracle WebLogic Server with SNMP* for information about using Simple Network Management Protocol (SNMP) to monitor WebLogic domains.

Refer to the Oracle Fusion Middleware library on the Oracle Help Center for additional information.

Conventions

The following text conventions are used in this document:

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

1

Oracle SOA Suite Custom WLST Commands

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 `SOA_HOME/common/bin`.

WLST provides both offline and online modes. Offline commands can be used without connecting to the Administration Server. When you first invoke WLST, you are in offline mode. You need to connect to the Administration server, using the `connect` command, before you can use the online mode commands.

This chapter includes the following sections:

- [Overview of WSLT Command Categories](#)
- [Deployment Commands](#)
- [WLST MDS Commands](#)
- [SOA Composite Application Management Commands](#)
- [Configuration Plan Management Commands](#)
- [Task Validation Commands](#)
- [SOA Composite Application Compilation Commands](#)
- [SOA Composite Application Packaging Commands](#)
- [SOA Composite Application Test Commands](#)
- [SOA Composite Application HTTP Client-Based Export and Import Commands](#)
- [SOA Composite Application MBean-Based Export and Import Commands](#)
- [SOA Composite Application Partition Management Commands](#)
- [SOA Composite Application Offline Deployment Management Commands](#)
- [SOA Health Check Commands](#)

For additional details about deployment, configuration plans, and test suites, see *Developing SOA Applications with Oracle SOA Suite*.

1.1 Overview of WSLT Command Categories

WLST commands are divided into the categories shown in [Table 1-1](#).

Table 1-1 Oracle SOA Suite Command Categories

Command category	Description
Deployment Commands	Deploy and undeploy SOA composite applications.
SOA Composite Application Management Commands	Start, stop, activate, retire, assign a default revision version, and list deployed SOA composite applications.
Configuration Plan Management Commands	Attach, extract, generate, and validate configuration plans for SOA composite applications.
Task Validation Commands	Validate human workflow tasks.
SOA Composite Application Compilation Commands	Compile SOA composite applications.
SOA Composite Application Packaging Commands	Package SOA composite applications into archive files to deploy.
SOA Composite Application Test Commands	Test SOA composite applications prior to deployment in a production environment.
SOA Composite Application HTTP Client-Based Export and Import Commands	Export and import SOA composite applications based on the HTTP client.
SOA Composite Application MBean-Based Export and Import Commands	Export and import SOA composite applications on the server-based composite store MBean (CompositeStoreMBean).
SOA Composite Application Partition Management Commands	Logically group different revisions of your SOA composite applications into separate partitions.
SOA Composite Application Offline Deployment Management Commands	Manage offline deployments of SOA composite applications and shared data.

1.2 Deployment Commands

Use the deployment commands, listed in [Table 1-2](#), to deploy and undeploy SOA composite applications.

Table 1-2 Deployment Commands for WLST Configuration

Use this command...	To...	Use with WLST...
sca_deployComposite	Deploy a SOA composite application.	Offline

Table 1-2 (Cont.) Deployment Commands for WLST Configuration

Use this command...	To...	Use with WLST...
sca_undeployComposite	Undeploy a SOA composite application.	Offline

1.2.1 sca_deployComposite

Command Category: Deployment Commands

Use with WLST: Offline

1.2.1.1 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 [SOA Composite Application Packaging Commands](#) for instructions on packaging a SOA composite application.

1.2.1.2 Syntax

```
sca_deployComposite(serverURL, sarLocation, [overwrite], [user], [password],
[forceDefault], [configplan], [partition] [keepInstancesOnRedeploy])
```

Argument	Definition
<i>serverURL</i>	URL of the server that hosts the SOA Infrastructure application (for example, <code>http://myhost10:7001</code>).
<i>sarLocation</i>	Absolute path to one the following: <ul style="list-style-type: none"> SOA archive (SAR) file. <p>A SAR file is a special JAR file that requires a prefix of <code>sca_</code> (for example, <code>sca_HelloWorld_rev1.0.jar</code>). The SAR file can be deployed with the deployment commands (such as <code>sca_deployComposite()</code>), but a regular <code>.jar</code> file is not treated as a special SAR file.</p> ZIP file that includes multiple SARs, metadata archives (MARs), or both.
<i>overwrite</i>	Optional. Indicates whether to overwrite an existing SOA composite application. <ul style="list-style-type: none"> <code>false</code> (default): Does not overwrite the application. <code>true</code>: Overwrites the application.
<i>user</i>	Optional. User name to access the composite deployer servlet when basic authentication is configured.
<i>password</i>	Optional. Password to access the composite deployer servlet when basic authentication is configured.
<i>forceDefault</i>	Optional. Indicates whether to set the new composite as the default. <ul style="list-style-type: none"> <code>true</code> (default): Makes it the default composite. <code>false</code>: Does not make it the default composite.

Argument	Definition
<i>configplan</i>	Optional. Absolute path of a configuration plan to be applied to a specified SAR file or to all SAR files included in the ZIP file.
<i>partition</i>	Optional. The name of the partition in which to deploy the SOA composite application. The default value is <code>default</code> . If you do not specify a partition, the composite is automatically deployed into the <code>default</code> partition.
<i>keepInstancesOnRedeploy</i>	Optional. Specifies whether to keep the instances on redeploy. The default value is <code>false</code> . This argument is used when BPM is installed.

 **Note:**

Human workflow artifacts such as task mapped attributes (previously known as flex field mappings) and rules (such as vacation rules) are defined based on the namespace of the task definition. Therefore, the following issues are true when the same SOA composite application with a human workflow task is deployed into multiple partitions:

- For the same task definition type, mapped attributes defined in one partition are visible in another partition.
- Rules defined on a task definition in one partition can apply to the same definition in another partition.

1.2.1.3 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")
```

The following example deploys the HelloWorld application to the myPartition partition.

```
wls:/mydomain/ServerConfig> sca_deployComposite("http://stadp10:7001",
"/tmp/sca_HelloWorld_rev1.0.jar", partition="myPartition")
```

1.2.2 sca_undeployComposite

Command Category: Deployment Commands

Use with WLST: Offline

1.2.2.1 Description

Undeploys a currently deployed SOA composite application.

1.2.2.2 Syntax

```
sca_undeployComposite(serverURL, compositeName, revision, [user], [password],
[partition])
```

Argument	Definition
<i>serverURL</i>	URL of the server that hosts the SOA Infrastructure application (for example, <code>http://myhost10:7001</code>).
<i>compositeName</i>	Name of the SOA composite application.
<i>revision</i>	Revision ID of the SOA composite application.
<i>user</i>	Optional. User name to access the composite deployer servlet when basic authentication is configured.
<i>password</i>	Optional. Password to access the composite deployer servlet when basic authentication is configured.
<i>partition</i>	Optional. The name of the partition in which the SOA composite application is located. The default value is <code>default</code> . If you do not specify a partition, the <code>default</code> partition is searched for the SOA composite application. However, no other partitions are searched.

1.2.2.3 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:
```

The following example undeploys the HelloWorld application in the myPartition partition.

```
wls:/mydomain/ServerConfig> sca_undeployComposite("http://stadb10:7001",
"HelloWorld", "1.0", partition='myPartition')
```

1.3 WLST MDS Commands

Use the WLST MDS Commands to import and export metadata.

Table 1-3 WLST MDS Commands for Import and Export

Use this command...	To...	Use with WLST...
importMetadata	Import metadata	Offline
exportMetadata	Export metadata	Offline

1.3.1 importMetadata

The application metadata can be transferred from one server location (testing) to another server location (production) by exporting and importing the metadata. Metadata can be imported using this command.

Syntax

```
importMetadata( application, server, fromLocation, docs, [restrictCustTo],
[excludeAllCust], [excludeBaseDocs], [excludeExtendedMetadata],
[cancelOnException], [applicationVersion])
```

Argument	Definition
application	Application name for which the metadata has to be imported
server	The target server on which this application is deployed
fromLocation	Source directory from where documents will be selected for transfer. fromLocation can be used as temporary file system location for transferring metadata from one server to another
docs	List of comma separated fully qualified document name(s) and/or document name patterns (* and ** patterns)

Argument	Definition
<code>restrictCustTo</code>	A list of Customization Layer names used to restrict the import operation to import customization documents that match the specified customization layers. This option will be ignored if <code>excludeAllCust</code> option is also specified
<code>excludeAllCust</code>	Specifies whether or not to import all customization documents. This option overrides the <code>restrictCustTo</code>
<code>excludeBaseDocs</code>	Specifies whether or not to import base documents
<code>excludeExtendedMetadata</code>	Specifies whether or not to import the Extended Metadata documents
<code>cancelOnException</code>	Whether or not to abort the import operation when an exception is encountered. On abort, the delete is rolled back if supported by the target store
<code>applicationVersion</code>	The application version in case multiple versions of the same application are deployed

Example

```
wls:/weblogic/serverConfig>importMetadata(application='sampleApp',
server='srg',
fromLocation='/tmp/myrepos',
docs='/**')
```

1.3.2 exportMetadata

The application metadata can be transferred from one server location (testing) to another server location (production) by exporting and importing the metadata. Metadata can be exported using this command.

Syntax

```
exportMetadata(application, server, toLocation, docs, [restrictCustTo],
[excludeAllCust], [excludeBaseDocs], [excludeExtendedMetadata],
[fromLabel], [toLabel], [applicationVersion])
```

Argument	Definition
<code>application</code>	Application name for which the metadata has to be exported

Argument	Definition
server	The target server on which this application is deployed
toLocation	Target directory to which documents selected from source partition will be transferred. toLocation can be used as temporary file system for transferring metadata from one server to another
docs	List of comma separated fully qualified document name(s) and/or document name patterns (* and ** patterns)
restrictCustTo	A list of Customization Layer names used to restrict the export operation to export customization documents that match the specified customization layers. This option will be ignored if excludeAllCust option is also specified
excludeAllCust	Specifies whether or not to export all customization documents. This option overrides the restrictCustTo
excludeBaseDocs	Specifies whether or not to export base documents
excludeExtendedMetadata	Specifies whether or not to export the Extended Metadata documents
fromLabel	If specified, transfers the documents from the source partition that is associated with this label
toLabel	If specified, will work with fromLabel variable to transfers the delta between fromLabel to toLabel from the source partition
applicationVersion	The application version in case multiple versions of the same application are deployed

Examples

To export the tip of all documents from a metadata repository's applications partition to a share filesystem:

```
wls:/weblogic/serverConfig>exportMetadata(application='sampleApp',
server='srg',
toLocation='/tmp/myrepos',
docs='/**')
```

To export only customizations for document /a.xml and /b.xml from application partition with label 'Oct2008':

```
wls:/weblogic/serverConfig>exportMetadata(application='sampleApp',
server='srg',
docs='/a.xml,/b.xml',
fromLabel='Oct2008'
)
```

1.4 SOA Composite Application Management Commands

Use the management commands, listed in [Table 1-4](#), to start, stop, activate, retire, assign a default revision version, and list deployed SOA composite applications.

Table 1-4 SOA Composite Application Management Commands for WLST Configuration

Use this command...	To...	Use with WLST...
sca_startComposite	Start a previously stopped SOA composite application.	Offline
sca_stopComposite	Stop a SOA composite application.	Offline
sca_activateComposite	Activate a previously retired SOA composite application.	Offline
sca_retireComposite	Retire a SOA composite application.	Offline
sca_assignDefaultComposite	Assign the default revision version to a SOA composite application.	Offline
sca_listDeployedComposites	List the deployed SOA composite applications.	Offline

1.4.1 sca_startComposite

Command Category: Application Management Commands

Use with WLST: Offline

1.4.1.1 Description

Starts a previously stopped SOA composite application.

1.4.1.2 Syntax

```
sca_startComposite(host, port, user, password, compositeName, revision, [label],
[partition])
```

Argument	Definition
<i>host</i>	Hostname of the Oracle WebLogic Server (for example, myhost).
<i>port</i>	Port of the Oracle WebLogic Server (for example, 7001).

Argument	Definition
<i>user</i>	User name for connecting to the running server to get MBean information (for example, <code>weblogic</code>).
<i>password</i>	Password for the user name.
<i>compositeName</i>	Name of the SOA composite application.
<i>revision</i>	Revision of the SOA composite application.
<i>label</i>	Optional. Label of the SOA composite application. The label identifies the metadata service (MDS) artifacts associated with the application. If the label is not specified, the system finds the latest one.
<i>partition</i>	Optional. The name of the partition in which the SOA composite application is located. The default value is <code>default</code> . If you do not specify a partition, the <code>default</code> partition is searched for the SOA composite application. However, no other partitions are searched.

1.4.1.3 Example

The following example starts revision 1.0 of the `HelloWorld` application.

```
wls:/mydomain/ServerConfig> sca_startComposite("myhost", "7001", "weblogic",
"welcome1", "HelloWorld", "1.0")
```

The following example starts revision 1.0 of the `HelloWorld` application in the partition `myPartition`.

```
wls:/mydomain/ServerConfig> sca_startComposite("stadp10", "7001", "weblogic",
"weblogic", "HelloWorld", "1.0", partition="myPartition")
```

1.4.2 sca_stopComposite

Command Category: Application Management Commands

Use with WLST: Offline

1.4.2.1 Description

Stops a currently running SOA composite application.

1.4.2.2 Syntax

```
sca_stopComposite(host, port, user, password, compositeName, revision, [label],
[partition])
```

Argument	Definition
<i>host</i>	Hostname of the Oracle WebLogic Server (for example, <code>myhost</code>).
<i>port</i>	Port of the Oracle WebLogic Server (for example, <code>7001</code>).
<i>user</i>	User name for connecting to the running server to get MBean information (for example, <code>weblogic</code>).
<i>password</i>	Password for the user name.

Argument	Definition
<i>compositeName</i>	Name of the SOA composite application.
<i>revision</i>	Revision of the SOA composite application.
<i>label</i>	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.
<i>partition</i>	Optional. The name of the partition in which the SOA composite application is located. The default value is <code>default</code> . If you do not specify a partition, the <code>default</code> partition is searched for the SOA composite application. However, no other partitions are searched.

1.4.2.3 Example

The following example stops revision 1.0 of the `HelloWorld` application.

```
wls:/mydomain/ServerConfig> sca_stopComposite("myhost", "7001", "weblogic",
"welcome1", "HelloWorld", "1.0")
```

The following example stops revision 1.0 of the `HelloWorld` application in the partition `myPartition`.

```
wls:/mydomain/ServerConfig> sca_stopComposite("stadp10", "7001", "weblogic",
"weblogic", "HelloWorld", "1.0", partition="myPartition")
```

1.4.3 sca_activateComposite

Command Category: Application Management Commands

Use with WLST: Offline

1.4.3.1 Description

Activates a retired SOA composite application. You can then create new instances.

1.4.3.2 Syntax

```
sca_activateComposite(host, port, user, password, compositeName, revision, [label],
[partition])
```

Argument	Definition
<i>host</i>	Hostname of the Oracle WebLogic Server (for example, <code>myhost</code>).
<i>port</i>	Port of the Oracle WebLogic Server (for example, <code>7001</code>).
<i>user</i>	User name for connecting to the running server to get MBean information (for example, <code>weblogic</code>).
<i>password</i>	Password for the user name.
<i>compositeName</i>	Name of the SOA composite application.
<i>revision</i>	Revision of the SOA composite application.

Argument	Definition
<i>label</i>	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.
<i>partition</i>	Optional. The name of the partition in which the SOA composite application is located. The default value is default. If you do not specify a partition, the default partition is searched for the SOA composite application. However, no other partitions are searched.

1.4.3.3 Example

The following example activates revision 1.0 of the HelloWorld application.

```
wls:/mydomain/ServerConfig> sca_activateComposite("myhost", "7001", "weblogic",
"welcome1", "HelloWorld", "1.0")
```

The following example activates revision 1.0 of the HelloWorld application in the partition myPartition.

```
wls:/mydomain/ServerConfig> sca_activateComposite("stadp10", "7001", "weblogic",
"weblogic", "HelloWorld", "1.0", partition="myPartition")
```

1.4.4 sca_retireComposite

Command Category: Application Management Commands

Use with WLST: Offline

1.4.4.1 Description

Retires a SOA composite application. If the process life cycle is retired, you cannot create a new instance. Existing instances are allowed to complete normally.

1.4.4.2 Syntax

```
sca_retireComposite(host, port, user, password, compositeName, revision, [label],
[partition])
```

Argument	Definition
<i>host</i>	Hostname of the Oracle WebLogic Server (for example, myhost).
<i>port</i>	Port of the Oracle WebLogic Server (for example, 7001).
<i>user</i>	User name for connecting to the running server to get MBean information (for example, weblogic).
<i>password</i>	Password for the user name.
<i>compositeName</i>	Name of the SOA composite application.
<i>revision</i>	Revision of the SOA composite application.
<i>label</i>	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.

Argument	Definition
<i>partition</i>	Optional. The name of the partition in which the SOA composite application is located. The default value is <code>default</code> . If you do not specify a partition, the <code>default</code> partition is searched for the SOA composite application. However, no other partitions are searched.

1.4.4.3 Example

The following example retires revision 1.0 of the HelloWorld application.

```
wls:/mydomain/ServerConfig> sca_retireComposite("myhost", "7001", "weblogic",
"welcome1", "HelloWorld", "1.0")
```

The following example retires revision 1.0 of the HelloWorld application in the partition myPartition.

```
wls:/mydomain/ServerConfig> sca_retireComposite("stadpl0", "7001", "weblogic",
"weblogic", "HelloWorld", "1.0", partition="myPartition")
```

1.4.5 sca_assignDefaultComposite

Command Category: Application Management Commands

Use with WLST: Offline

1.4.5.1 Description

Sets a SOA composite application revision as the default version. This revision is instantiated when a new request comes in.

1.4.5.2 Syntax

```
sca_assignDefaultComposite(host, port, user, password, compositeName, revision,
[partition])
```

Argument	Definition
<i>host</i>	Hostname of the Oracle WebLogic Server (for example, myhost).
<i>port</i>	Port of the Oracle WebLogic Server (for example, 7001).
<i>user</i>	User name for connecting to the running server to get MBean information (for example, weblogic).
<i>password</i>	Password for the user name.
<i>compositeName</i>	Name of the SOA composite application.
<i>revision</i>	Revision of the SOA composite application.
<i>partition</i>	Optional. The name of the partition in which the SOA composite application is located. The default value is <code>default</code> . If you do not specify a partition, the <code>default</code> partition is searched for the SOA composite application. However, no other partitions are searched.

1.4.5.3 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")
```

The following example sets revision 1.0 of the HelloWorld application located in the partition myPartition as the default version.

```
wls:/mydomain/ServerConfig> sca_assignDefaultComposite("stadp10", "7001",  
"weblogic", "weblogic", "HelloWorld", "1.0", partition="myPartition")
```

1.4.6 sca_listDeployedComposites

Command Category: Application Management Commands

Use with WLST: Offline

1.4.6.1 Description

Lists all SOA composite applications deployed to the SOA platform.

1.4.6.2 Syntax

```
sca_listDeployedComposites(host, port, user, password)
```

Argument	Definition
<i>host</i>	Hostname of the Oracle WebLogic Server (for example, myhost).
<i>port</i>	Port of the Oracle WebLogic Server (for example, 7001).
<i>user</i>	User name for connecting to the running server to get MBean information (for example, weblogic).
<i>password</i>	Password for the user name.

1.4.6.3 Example

The following example lists all the deployed SOA composite applications on the server myhost.

```
wls:/mydomain/ServerConfig> sca_listDeployedComposites('myhost', '7001',  
'weblogic', 'welcome1')
```

1.5 Configuration Plan Management Commands

Use the configuration plan management commands, listed in [Table 1-5](#), to attach, extract, generate, and validate configuration plans for SOA composite applications.

Table 1-5 Configuration Plan Management Commands for WLST Configuration

Use this command...	To...	Use with WLST...
<code>sca_attachPlan</code>	Attach the configuration plan file to the SOA composite application JAR file.	Offline
<code>sca_extractPlan</code>	Extract a configuration plan packaged with the JAR file for editing.	Offline
<code>sca_generatePlan</code>	Generate a configuration plan for editing.	Offline
<code>sca_validatePlan</code>	Validate the configuration plan.	Offline

1.5.1 sca_attachPlan

Command Category: Configuration Plan Management Commands

Use with WLST: Offline

1.5.1.1 Description

Attaches the configuration plan file to the SOA composite application file. If a plan already exists in the file, it is not overwritten by default.

1.5.1.2 Syntax

```
sca_attachPlan(sar, configPlan, [overwrite], [verbose])
```

Argument	Definition
<i>sar</i>	Absolute path of the SAR file.
<i>configPlan</i>	Absolute path of the configuration plan file.
<i>overwrite</i>	Optional. Indicates whether to overwrite an existing configuration plan in the SAR file. <ul style="list-style-type: none"> • <code>false</code> (default): Does not overwrite the plan. • <code>true</code>: Overwrites the plan.
<i>verbose</i>	Optional. Indicates whether to print more information about the configuration plan attachment. <ul style="list-style-type: none"> • <code>true</code> (default): Prints more information. • <code>false</code>: Does not print more information.

1.5.1.3 Examples

The following example attaches the `configplan.xml` configuration plan file to the HelloWorld application.

```
wls:/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.

```
wls:/mydomain/ServerConfig> sca_attachPlan("/tmp/sca_HelloWorld_rev1.0.jar", "/tmp/  
configplan.xml", overwrite=true)
```

1.5.2 sca_extractPlan

Command Category: Configuration Plan Management Commands

Use with WLST: Offline

1.5.2.1 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`.

1.5.2.2 Syntax

```
sca_extractPlan(sar, configPlan, [overwrite], [verbose])
```

Argument	Definition
<i>sar</i>	Absolute path of a SAR file.
<i>configPlan</i>	Absolute path of a configuration plan file to which to be extracted.
<i>overwrite</i>	Optional. Indicates whether to overwrite the configuration file specified by <i>configPlan</i> , if it exists. <ul style="list-style-type: none">• false (default): Does not overwrite the plan.• true: Overwrites the plan.
<i>verbose</i>	Optional. Indicates whether to print more information about configuration plan extraction. <ul style="list-style-type: none">• true (default): Prints more information.• false: Does not print more information.

1.5.2.3 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)
```

1.5.3 sca_generatePlan

Command Category: Configuration Plan Management Commands

Use with WLST: Offline

1.5.3.1 Description

Generates a configuration plan for editing.

1.5.3.2 Syntax

```
sca_generatePlan(configPlan, sar, composite, [overwrite], [verbose])
```

Argument	Definition
<i>configPlan</i>	Absolute path of the configuration plan file to be generated. Either <i>configPlan</i> or <i>sar</i> is required. You cannot specify both.
<i>sar</i>	Absolute path of the SAR file. Either <i>configPlan</i> or <i>sar</i> is required. You cannot specify both.
<i>composite</i>	Absolute path of the <i>composite.xml</i> file in the expanded (unzipped) SAR directory.
<i>overwrite</i>	Optional. Indicates whether to overwrite an existing configuration plan file: <ul style="list-style-type: none"> • <code>false</code> (default): Does not overwrite the plan. • <code>true</code>: Overwrites the plan.
<i>verbose</i>	Indicates whether to print more information about plan generation: <ul style="list-style-type: none"> • <code>true</code> (default): Prints more information. • <code>false</code>: Does not print more information.

1.5.3.3 Examples

The following example generates the *myplan.xml* configuration plan file for the HelloWorld application.

```
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.

```
wls:/mydomain/ServerConfig> sca_generatePlan("/tmp/myplan2.xml",
composite="/tmp/HelloWorld_rev1.0/composite.xml", overwrite=true)
```

1.5.4 sca_validatePlan

Command Category: Configuration Plan Management Commands

Use with WLST: Offline

1.5.4.1 Description

Validates the configuration plan. This command identifies all search and replacement changes to be made during deployment. Use this option for debugging only.

1.5.4.2 Syntax

```
sca_validatePlan(reportFile, configPlan, [sar], [composite], [overwrite], [verbose])
```

Argument	Definition
<i>reportFile</i>	Absolute path of the report file to be generated. Validation results are written to this file.
<i>configPlan</i>	Absolute path of the configuration plan file.
<i>sar</i>	Optional. The absolute path of the SAR file.
<i>composite</i>	Optional. The absolute path of the <code>composite.xml</code> file in the expanded (unzipped) SAR directory.
<i>overwrite</i>	Optional. Indicates whether to overwrite an existing configuration plan file: <ul style="list-style-type: none"> <code>false</code> (default): Does not overwrite the plan. <code>true</code>: Overwrites the plan.
<i>verbose</i>	Optional. Indicates whether to print more information about configuration plan validation. <ul style="list-style-type: none"> <code>true</code> (default): Prints more information. <code>false</code>: Does not print more information.

1.5.4.3 Examples

The following example validates the `configplan.xml` configuration plan file for the HelloWorld application.

```
wls:/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)
```

1.6 Task Validation Commands

Use the task validation command, listed in [Table 1-6](#), to validate human workflow tasks.

Table 1-6 Task Validation Command for WLST Configuration

Use this command...	To...	Use with WLST...
sca_validateTask	Validate a human workflow task.	Offline

1.6.1 sca_validateTask

Command Category: Task Validation Commands

Use with WLST: Offline

1.6.1.1 Description

Validates a human workflow task contained in the `.task` file that you created when designing a human task in the Human Task Editor.

1.6.1.2 Syntax

```
sca_validateTask(taskFile, outXml, [displayLevel])
```

Argument	Definition
<i>taskFile</i>	Absolute path to the task definition file (<code>.task</code>).
<i>outXml</i>	Absolute path to an output XML file.
<i>displayLevel</i>	Optional. The level of information to display. The values can be 1, 2, or 3. The default value is 1.

1.6.1.3 Example

The following example validates the `WFTaskDefinition.task` file of the human task.

```
wls:/mydomain/ServerConfig> sca_validateTask("/tmp/WFTaskDefinition.task",
"/tmp/out.xml", displayLevel=2)
```

1.7 SOA Composite Application Compilation Commands

Use the compilation commands, listed in [Table 1-7](#), to compile SOA composite applications.

Table 1-7 SOA Composite Application Compilation Commands for WLST Configuration

Use this command...	To...	Use with WLST...
sca_setProp	Set JVM system properties.	Offline
sca_compile	Compile a SOA composite application.	Offline

1.7.1 sca_setProp

Command Category: Application Compilation Commands

Use with WLST: Offline

1.7.1.1 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.

1.7.1.2 Syntax

```
sca_setProp(propName, propValue)
```

Argument	Definition
<i>propName</i>	Property name.
<i>propValue</i>	Property value.

1.7.1.3 Example

The following example sets the property name and property value.

```
wls:/mydomain/ServerConfig> sca_setProp("oracle.home",
"/scratch/myusername/beahome/AS11gR1SOA")
```

1.7.2 sca_compile

Command Category: Application Compilation Commands

Use with WLST: Offline

1.7.2.1 Description

Compiles a SOA composite application.



Note:

The `sca_compile` command requires the `oracle.home` property to find the `ant-sca-compile.xml` script. This must be set once. You can use the `scac_setProp` command or the `oracleHome` property to set a value.

1.7.2.2 Syntax

```
sca_compile(composite, [outXml], [error], [appHome], [displayLevel], [oracleHome])
```

Argument	Definition
<i>composite</i>	Absolute path of a composite file in the expanded (unzipped) SAR directory.
<i>outXml</i>	Optional. Absolute path of an output XML file.
<i>error</i>	Optional. Absolute path of an error file.
<i>appHome</i>	Optional only if you are compiling from inside the project directory. Absolute path of the application home directory. This property is required if you have shared data.
<i>displayLevel</i>	Optional. The level of information to display. The default value is 1.

Argument	Definition
<i>oracleHome</i>	Optional. The <code>oracle.home</code> property.
<i>reportSchemaValidationErrors</i>	Optional. Indicates whether schema validation error is reported. The default value is false.

1.7.2.3 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")
```

1.8 SOA Composite Application Packaging Commands

Use the packaging command, listed in [Table 1-8](#), to package SOA composite applications into a composite SAR file.

Table 1-8 SOA Composite Application Packaging Command for WLST Configuration

Use this command...	To...	Use with WLST...
sca_package	Package the SOA composite application files into a composite SAR file.	Offline

1.8.1 sca_package

Command Category: Application Packaging Commands

Use with WLST: Offline

1.8.1.1 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.

1.8.1.2 Syntax

```
sca_package(compositeDir, compositeName, revision, [appHome], [oracleHome])
```

Argument	Definition
<code>compositeDir</code>	Absolute path of a directory that contains composite artifacts.
<code>compositeName</code>	Name of the composite.
<code>revision</code>	Revision ID of the composite.
<code>appHome</code>	Optional. Absolute path of the application home directory. This property is required if you have shared data.
<code>oracleHome</code>	Optional. The <code>oracle.home</code> property.

1.8.1.3 Examples

The following example packages the `OrderBookingComposite` application. The `appHome` property is set because this application uses shared data.

```
wls:/mydomain/ServerConfig> sca_package("/tmp/app_data/OrderBookingComposite",
"OrderBookingComposite", "1.0", appHome="/tmp/app_data")
```

The following example packages the `HelloSOAComposite` application.

```
wls:/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.

```
wls:/mydomain/ServerConfig> sca_package
("/tmp/HelloSOAApplication/HelloSOAComposite", "HelloSOAComposite", "1.0",
oracleHome="/scratch/myusername/beahome/AS11gR1SOA")
```

1.9 SOA Composite Application Test Commands

Use the SOA composite application test command, listed in [Table 1-9](#), to test a SOA composite applications.

Table 1-9 SOA Composite Application Test Command for WLST Configuration

Use this command...	To...	Use with WLST...
sca_test	Test deployed SOA composite applications.	Offline

1.9.1 sca_test

Command Category: Application Test Commands

Use with WLST: Offline

1.9.1.1 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`.

1.9.1.2 Syntax

```
sca_test('compositeName', 'revision', 'testsuiteName', 'jndiPropFile',
[oracleHome='oracleHome'], [javaHome='javaHome'])
```

Argument	Definition
<i>compositeName</i>	Name of the SOA composite application.
<i>revision</i>	Revision ID of the SOA composite application.
<i>testsuiteName</i>	Name of the test suite.
<i>jndiPropFile</i>	Absolute path to the JNDI property file.
<i>oracleHome</i>	Optional. The <code>oracle.home</code> system property.
<i>javaHome</i>	Optional. The <code>java.passed.home</code> system property.

1.9.1.3 Examples

The following example runs the `OrderBookingMainTestsuite` test suite.

```
wls:/mydomain/ServerConfig> sca_test('OrderBookingComposite', '1.0',
'OrderBookingMainTestsuite', '/tmp/tmp-jndi.properties',
oracleHome='/scratch/<user>/beahome/AS11gR1SOA/',
javaHome='/scratch/<user>/beahome/jdk160_05')
```

1.10 SOA Composite Application HTTP Client-Based Export and Import Commands

Use the SOA composite application commands, listed in [Table 1-10](#), to export and import SOA composite applications based on the HTTP client. The SOA Infrastructure must be running to use these commands.

Table 1-10 SOA Composite Application Export and Import Commands for WLST Configuration

Use this command...	To...	Use with WLST...
sca_exportComposite	Export a SOA composite application into a SAR file.	Offline
sca_exportUpdates	Export postdeployment changes of a SOA composite application into a JAR file.	Offline
sca_importUpdates	Import postdeployment changes of a SOA composite application.	Offline
sca_exportSharedData	Export shared data of a given pattern into a JAR file.	Offline
sca_removeSharedData	Removes a top-level shared data folder.	Offline

1.10.1 sca_exportComposite

Command Category: Application Export and Import Commands

Use with WLST: Offline

1.10.1.1 Description

Exports a SOA composite application into a SAR file.

1.10.1.2 Syntax

```
sca_exportComposite(serverURL, updateType, sarFile, compositeName, revision,
[user], [password], [partition])
```

Argument	Definition
<i>serverURL</i>	URL of the server that hosts the SOA Infrastructure application (for example, <code>http://stabc:8001</code>).
<i>updateType</i>	Type of postdeployment changes to be exported: <ul style="list-style-type: none"> • <code>all</code>: Includes all postdeployment changes. • <code>property</code>: Includes only property postdeployment changes (binding component properties, composite properties such as audit level settings and payload validation status, and policy attachments). • <code>runtime</code>: Includes only runtime (rules dictionary and domain value maps (DVMs)) and metadata postdeployment changes. • <code>none</code>: Exports the original composite without any postdeployment changes (including property changes and runtime changes).
<i>sarFile</i>	Absolute path of a SAR file to generate (a <code>.jar</code> file that begins with <code>sca_</code>).
<i>compositeName</i>	Name of the composite to export.
<i>revision</i>	Revision of the composite to export.
<i>user</i>	Optional. The user name for accessing the server when basic configuration is configured. Use the following syntax for this argument: <code>user='username'</code>

Argument	Definition
<i>password</i>	Optional. The password for accessing the server when basic configuration is configured. Use the following syntax for this argument: <code>password='password'</code>
<i>partition</i>	Optional. The name of the partition in which the SOA composite application is located. The default value is default.

1.10.1.3 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')
```

The following example exports a composite in the `myPartition` partition without including any postdeployment updates:

```
wls:/offline/mydomain/ServerConfig> sca_exportComposite('http://stabc:8001',
'none', '/tmp/sca>HelloWorld_rev1.0.jar', 'HelloWorld', '1.0',
partition='myPartition')
```

1.10.2 sca_exportUpdates

Command Category: Application Export and Import Commands

Use with WLST: Offline

1.10.2.1 Description

Exports postdeployment changes of a SOA composite application into a JAR file.

1.10.2.2 Syntax

```
sca_exportUpdates(serverURL, updateType, jarFile, compositeName, revision,
[user], [password], [partition])
```

Argument	Definition
<i>serverURL</i>	URL of the server that hosts the SOA Infrastructure application (for example, <code>http://stabc:8001</code>).
<i>updateType</i>	The type of postdeployment changes to be exported. <ul style="list-style-type: none"> • <code>all</code>: Includes all postdeployment changes. • <code>property</code>: Includes only property postdeployment changes (binding component properties, composite properties such as audit level settings and payload validation status, and policy attachments). • <code>runtime</code>: Includes only runtime (rules dictionary and domain value maps (DVMs)) and metadata postdeployment changes.
<i>jarFile</i>	Absolute path of a JAR file to generate. <code>sca_exportUpdates()</code> creates a regular <code>.jar</code> file that cannot be imported using regular deployment commands. It must be imported by using <code>sca_importUpdates()</code> .
<i>compositeName</i>	Name of the composite to export.
<i>revision</i>	Revision of the composite to export.
<i>user</i>	Optional. The user name for accessing the server when basic configuration is configured. Use the following syntax for this argument: <code>user='username'</code>
<i>password</i>	Optional. The password for accessing the server when basic configuration is configured. Use the following syntax for this argument: <code>password='password'</code>
<i>partition</i>	Optional. The name of the partition in which the SOA composite application is located. The default value is <code>default</code> .

1.10.2.3 Examples

The following example exports all postdeployment updates.

```
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.

```
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.

```
wls:/offline/mydomain/ServerConfig> sca_exportUpdates('http://stabc:8001',
'runtime', '/tmp/runtime-HelloWorld_rev1.0.jar', 'HelloWorld', '1.0')
```

The following example exports postdeployment changes of a composite in the partition `myPartition` into a JAR file.

```
wls:/offline/mydomain/ServerConfig> sca_exportUpdates('http://stabc:8001',
'runtime', '/tmp/runtime-HelloWorld_rev1.0.jar', 'HelloWorld', '1.0',
partition='myPartition')
```

1.10.3 sca_importUpdates

Command Category: Application Export and Import Commands

Use with WLST: Offline

1.10.3.1 Description

Imports postdeployment changes of a SOA composite application.

1.10.3.2 Syntax

```
sca_importUpdates(serverURL, jarFile, compositeName, revision, [user],
[password])
```

Argument	Definition
<i>serverURL</i>	URL of the server that hosts the SOA Infrastructure application (for example, <code>http://stabc:8001</code>).
<i>jarFile</i>	Absolute path of a JAR file that contains postdeployment changes.
<i>compositeName</i>	Name of the composite to which the postdeployment changes are imported.
<i>revision</i>	Revision of the composite to which the postdeployment changes are imported.
<i>user</i>	Optional. The user name for accessing the server when basic configuration is configured. Use the following syntax for this argument: <code>user='username'</code>
<i>password</i>	Optional. The password for accessing the server when basic configuration is configured. Use the following syntax for this argument: <code>password='password'</code>
<i>partition</i>	Optional. The name of the partition in which the SOA composite application is located. The default value is default.

1.10.3.3 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')
```

The following example imports postdeployment changes of a composite in the partition myPartition.

```
wls:/offline/mydomain/ServerConfig> sca_importUpdates('http://stabc:8001',
'/tmp/all-HelloWorld_rev1.0.jar', 'HelloWorld', '1.0', partition='myPartition')
```

1.10.4 sca_exportSharedData

Command Category: Application Export and Import Commands

Use with WLST: Offline

1.10.4.1 Description

Exports shared data of a given pattern into a JAR file.

1.10.4.2 Syntax

```
sca_exportSharedData(serverURL, jarFile, pattern, [user], [password])
```

Argument	Definition
<i>serverURL</i>	URL of the server that hosts the SOA Infrastructure application (for example, <code>http://stabc:8001</code>).
<i>jarFile</i>	Absolute path of a JAR file to generate.
<i>pattern</i>	The file pattern supported by MDS transfer APIs. Use the semicolon delimiter (<code>:</code>) if more than one pattern is specified. Exclude the shared data namespace <code>/apps</code> in the pattern. For example: <code>/Project1/**;/Project2/**</code> This example exports all documents under <code>/apps/Project1</code> and <code>/apps/Project2</code> .
<i>user</i>	Optional. The user name for accessing the server when basic configuration is configured. Use the following syntax for this argument: <code>user='username'</code>
<i>password</i>	Optional. The password for accessing the server when basic configuration is configured. Use the following syntax for this argument: <code>password='password'</code>

1.10.4.3 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/**')
```

1.10.5 sca_removeSharedData

Command Category: Application Export and Import Commands

Use with WLST: Offline

1.10.5.1 Description

Removes a top-level shared data folder, even if there are composites deployed in the service engine.

1.10.5.2 Syntax

```
sca_removeSharedData(serverURL, folderName, [user], [password])
```


Argument	Definition
<i>serverURL</i>	URL of the server that hosts the SOA Infrastructure application (for example, <code>http://stabc:8001</code>).
<i>folderName</i>	The name of a top-level shared data folder to be removed.
<i>user</i>	Optional. The user name for accessing the server when basic configuration is configured. Use the following syntax for this argument: <code>user='username'</code>
<i>password</i>	Optional. The password for accessing the server when basic configuration is configured. Use the following syntax for this argument: <code>password='password'</code>

1.10.5.3 Examples

The following example removes the top-level shared data `Project1` folder.

```
sca_removeSharedData('http://stabc:8001', 'Project1')
```

1.11 SOA Composite Application MBean-Based Export and Import Commands

Use the deployment commands, listed in [Table 1-11](#), to export and import SOA composite applications on the server-based composite store MBean (`CompositeStoreMXBean`).

Table 1-11 SOA Composite Application Export and Import Commands for WLST Configuration

Use this command...	To...	Use with WLST...
sca_exportCompositeMb	Export a SOA composite application into a SAR file.	Online
sca_exportUpdatesMb	Export postdeployment changes of a SOA composite application into a JAR file.	Online
sca_importUpdatesMb	Import postdeployment changes of a SOA composite application.	Online
sca_exportSharedDataMb	Export shared data of a given pattern into a JAR file.	Online

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()
```

1.11.1 sca_exportCompositeMb

Command Category: Application Export and Import Commands

Use with WLST: Online

1.11.1.1 Description

Exports a SOA composite application into a SAR file.

1.11.1.2 Syntax

```
sca_exportCompositeMb(updateType, sarFile, compositeName, revision)
```

Argument	Definition
<i>updateType</i>	Type of postdeployment changes to be exported: <ul style="list-style-type: none"> • <code>all</code>: All postdeployment changes are included. • <code>property</code>: Property changes are included (binding component properties, composite properties such as audit level settings and payload validation status, and policy attachments). • <code>runtime</code>: Postdeployment runtime changes are included (rules dictionary and domain value maps (DVMs)).
<i>sarFile</i>	Absolute path of a SAR file to generate.
<i>compositeName</i>	Name of the composite to export.
<i>revision</i>	Revision of the composite to export.

1.11.1.3 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')
```

1.11.2 sca_exportUpdatesMb

Command Category: Application Export and Import Commands

Use with WLST: Online

1.11.2.1 Description

Exports postdeployment changes of a SOA composite application into a JAR file.

1.11.2.2 Syntax

```
sca_exportUpdatesMb(updateType, jarFile, compositeName, revision)
```

Argument	Definition
<i>updateType</i>	Type of postdeployment changes to be exported: all, property, or runtime.
<i>jarFile</i>	Absolute path of a JAR file to generate.
<i>compositeName</i>	Name of the composite to export.
<i>revision</i>	Revision of the composite to export.

1.11.2.3 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')
```

1.11.3 sca_importUpdatesMb

Command Category: Application Export and Import Commands

Use with WLST: Online

1.11.3.1 Description

Imports postdeployment changes of a SOA composite application.

1.11.3.2 Syntax

```
sca_importUpdatesMb(jarFile, compositeName, revision)
```

Argument	Definition
<i>jarFile</i>	Absolute path of a JAR file that contains postdeployment changes.
<i>compositeName</i>	Name of the composite to which the postdeployment changes are imported.
<i>revision</i>	Revision of the composite to which the postdeployment changes are imported.

1.11.3.3 Examples

The following example imports postdeployment changes of a SOA composite application.

```
wls:/mydomain/ServerConfig> sca_importUpdatesMb('/tmp/all-HelloWorld_rev1.0.jar',
'HelloWorld', '1.0')
```

1.11.4 sca_exportSharedDataMb

Command Category: Application Export and Import Commands

Use with WLST: Online

1.11.4.1 Description

Exports shared data of a given pattern into a JAR file.

1.11.4.2 Syntax

```
sca_exportSharedDataMb(jarFile, pattern)
```

Argument	Definition
<i>jarFile</i>	Absolute path of a JAR file to generate.
<i>pattern</i>	The file pattern supported by MDS transfer APIs. Use the semicolon (;) 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.

1.11.4.3 Examples

This example exports shared data of given pattern into a JAR file.

```
wls:/mydomain/ServerConfig> sca_exportSharedDataMb('/tmp/MySharedData.jar',
'/Project1/**')
```

1.12 SOA Composite Application Partition Management Commands

Use the deployment commands, listed in [Table 1-12](#), to manage partitions. Partitioning enable you to logically group different revisions of your SOA composite applications into separate sections. This is similar to the concept of domains in the 10.1.x releases of Oracle BPEL Process Manager.

Table 1-12 SOA Composite Application Partition Management Commands for WLST Configuration

Use this command...	To...	Use with WLST...
sca_createPartition	Create a partition.	Online
sca_deletePartition	Undeploy all SOA composite applications in a partition before deleting the partition.	Online
sca_startCompositesInPartition	Start all SOA composite applications in a partition.	Online
sca_stopCompositesInPartition	Stop all SOA composite applications in a partition.	Online
sca_activateCompositesInPartition	Activate all SOA composite applications in a partition.	Online
sca_retireCompositesInPartition	Retire all SOA composite applications in a partition.	Online
sca_listPartitions	List all partitions in the SOA Infrastructure.	Online
sca_listCompositesInPartition	List all composites in a specific partition.	Online
sca_createPartitionWMSG	Create a partition and associate it with a work manager group.	Online

1.12.1 sca_createPartition

Command Category: Application Partition Management Commands

Use with WLST: Online

1.12.1.1 Description

Creates a partition.

1.12.1.2 Syntax

```
sca_createPartition(partitionName)
```

Argument	Definition
<i>partitionName</i>	The name of the partition.

1.12.1.3 Examples

This example creates a partition named `myPartition`.

```
wls:/mydomain/ServerConfig> sca_createPartition('myPartition')
```

1.12.2 sca_deletePartition

Command Category: Application Partition Management Commands

Use with WLST: Online

1.12.2.1 Description

Undeploys all composites in a partition before deleting the partition.

1.12.2.2 Syntax

```
sca_deletePartition(partitionName)
```

Argument	Definition
<i>partitionName</i>	The name of the partition.

1.12.2.3 Examples

This example undeploys all composites in the `myPartition` partition before deleting the partition.

```
wls:/mydomain/ServerConfig> sca_deletePartition('myPartition')
```

1.12.3 sca_startCompositesInPartition

Command Category: Application Partition Management Commands

Use with WLST: Online

1.12.3.1 Description

Starts all composites in a partition.

1.12.3.2 Syntax

```
sca_startCompositesInPartition(partitionName)
```

Argument	Definition
<i>partitionName</i>	The name of the partition.

1.12.3.3 Examples

This example starts all composites in the `myPartition` partition.

```
wls:/mydomain/ServerConfig> sca_startCompositesInPartition('myPartition')
```

1.12.4 sca_stopCompositesInPartition

Command Category: Application Partition Management Commands

Use with WLST: Online

1.12.4.1 Description

Stops all composites in a partition.

1.12.4.2 Syntax

```
sca_stopCompositesInPartition(partitionName)
```

Argument	Definition
<i>partitionName</i>	The name of the partition.

1.12.4.3 Examples

This example stops all composites in the myPartition partition.

```
wls:/mydomain/ServerConfig> sca_stopCompositesInPartition('myPartition')
```

1.12.5 sca_activateCompositesInPartition

Command Category: Application Partition Management Commands

Use with WLST: Online

1.12.5.1 Description

Activates all composites in a partition.

1.12.5.2 Syntax

```
sca_activateCompositesInPartition(partitionName)
```

Argument	Definition
<i>partitionName</i>	The name of the partition.

1.12.5.3 Examples

This example activates all composites in the myPartition partition.

```
wls:/mydomain/ServerConfig> sca_activateCompositesInPartition('myPartition')
```

1.12.6 sca_retireCompositesInPartition

Command Category: Application Partition Management Commands

Use with WLST: Online

1.12.6.1 Description

Retires all composites in a partition.

1.12.6.2 Syntax

```
sca_retireCompositesInPartition(partitionName)
```

Argument	Definition
<i>partitionName</i>	The name of the partition.

1.12.6.3 Examples

This example retires all composites in the myPartition partition.

```
wls:/mydomain/ServerConfig> sca_retireCompositesInPartition('myPartition')
```

1.12.7 sca_listPartitions

Command Category: Application Partition Management Commands

Use with WLST: Online

1.12.7.1 Description

Lists all partitions in the SOA Infrastructure.

1.12.7.2 Syntax

```
sca_listPartitions()
```

1.12.7.3 Examples

This example lists all partitions in the SOA Infrastructure.

```
wls:/mydomain/ServerConfig> sca_listPartitions()
```

1.12.8 sca_listCompositesInPartition

Command Category: Application Partition Management Commands

Use with WLST: Online

1.12.8.1 Description

Lists all composites in a partition.

1.12.8.2 Syntax

```
sca_listCompositesInPartition(partitionName)
```


Argument	Definition
<i>partitionName</i>	The name of the partition.

1.12.8.3 Examples

This example lists all composites in the `myPartition` partition.

```
sca_listCompositesInPartition(myPartition)
```

1.12.9 sca_createPartitionWMG

Command Category: Application Partition Management Commands

Use with WLST: Online

1.12.9.1 Description

Creates a partition and associates it with a work manager group. The work manager group is created if it does not exist.

1.12.9.2 Syntax

```
sca_createPartitionWMG(partitionName, workmanagerGroupName,
workmanagerGroupDescription)
```

Argument	Definition
<i>partitionName</i>	The name of the partition to be created.
<i>workmanagerGroupName</i>	The name of the work manager group to use. The work manager group is created if it does not exist.
<i>workmanagerGroupDescription</i>	The description of the work manager group if it is to be created.

1.12.9.3 Examples

This example creates a partition named `myPartition` and associates it with a new work manager group named `myWMG`. Since this work manager group is being created, a description is provided.

```
sca_createPartitionWMG('myPartition', 'myWMG', 'my new workmanagerGroup')
```

1.13 SOA Composite Application Offline Deployment Management Commands

Use the deployment commands, listed in [Table 1-13](#), to manage offline deployments of SOA composite applications and shared data in the following file:

```
$domain/config/fmwconfig/composite-offline-deployments-default.xml
```

When you restart the SOA server, the SOA composite applications and shared data registered in the file are deployed.

Table 1-13 SOA Composite Application Offline Deployment Management Commands for WLST Configuration

Use this command...	To...	Use with WLST...
sca_registerCompositeOfflineDeployment	Register offline deployments of SOA composite applications and shared data.	Offline
sca_unregisterCompositeOfflineDeployment	Unregister offline deployments of SOA composite applications and shared data	Offline

Note the following guidelines when using these WLST commands:

- If the same SOA composite application is registered again and the corresponding SAR file has the same name, the composite is not redeployed. The composite is bypassed during offline deployment. This is because offline deployment does not support composite redeployments.
- Ensure that your SOA composite application JAR name is prefixed with `sca_`. The `sca_registerCompositeOfflineDeployment` command registers either shared data deployment or composite deployment. The composite SAR file is *always* named as `sca_*.jar`. Only JAR files that start with `sca_` are considered composite SAR files. JAR files that do not start with `sca_` are treated as a shared data file.
- When you register a newer revision of a SOA composite application (for example, revision 2.0 of `sca_HelloWorld.jar`) with the `sca_registerCompositeOfflineDeployment` command and then restart the server, the previous default SOA composite application revision (for example, revision 1.0 of `sca_HelloWorld.jar`) is retired. Any runtime changes that were performed in the older 1.0 revision with a runtime tool such as Oracle SOA Composer are merged into the newer 2.0 revision.

For information about offline deployment with configuration files, see Section "Deploying SOA Composite Applications with No Servers Running" of *Developing SOA Applications with Oracle SOA Suite*.

For more information about SAR file naming conventions, see Section "Deployed Service Archives" of *Developing SOA Applications with Oracle SOA Suite*.

1.13.1 `sca_registerCompositeOfflineDeployment`

Command Category: Application Offline Deployment Management Commands

Use with WLST: Offline

1.13.1.1 Description

Registers offline deployments of SOA composite applications and shared data in the `composite-offline-deployments-default.xml` file. Registration enables the SOA composite applications and shared data listed in the file to be deployed when the SOA server is restarted.

1.13.1.2 Syntax

```
sca_registerCompositeOfflineDeployment(domainDir, fileLocation, partition)
```

Argument	Definition
<i>domainDir</i>	The absolute path to the server domain.
<i>fileLocation</i>	The absolute path of a composite SAR file or shared data JAR file.
<i>default</i>	The partition in which to deploy the composite, This argument is optional. If not specified, the default partition named <i>default</i> is used.

1.13.1.3 Examples

This example specifies the domain directory and the file location for the `sca_HelloWorld.jar` composite.

```
wls:/mydomain/ServerConfig> sca_registerCompositeOfflineDeployment
('/scratch/aimel/fmwhome12/user_projects/domains/soainfra',
'/tmp/sca_HelloWorld.jar')
```

This example specifies the domain directory, the file location for the `sca_App1.jar` composite, and the partition `myPartition`.

```
wls:/mydomain/ServerConfig> sca_registerCompositeOfflineDeployment
('/scratch/aimel/fmwhome12/user_projects/domains/soainfra', '/tmp/sca_App1.jar',
partition='myPartition')
```

This example specifies the domain directory and the file location for the `shareddata.jar` shared data file.

```
wls:/mydomain/ServerConfig> sca_registerCompositeOfflineDeployment
('/scratch/aimel/fmwhome12/user_projects/domains/soainfra',
'/tmp/shareddata.jar')
```

1.13.2 sca_unregisterCompositeOfflineDeployment

Command Category: Application Offline Deployment Management Commands

Use with WLST: Offline

1.13.2.1 Description

Unregisters (removes) the SOA composite application or shared data from the `$domain/config/fmwconfig/composite-offline-deployments-default.xml` file so that it is not processed during a server restart. The composite or shared data is not checked during the offline deployment process. This does not impact the existing SOA composite applications. It does not undeploy or retire the composite when the server restarts.

1.13.2.2 Syntax

```
sca_unregisterCompositeOfflineDeployment(domainDir, fileLocation)
```

Argument	Definition
<i>domainDir</i>	The absolute path to the server domain.
<i>fileLocation</i>	The absolute path of a composite SAR file or a shared data JAR file.

1.13.2.3 Examples

This example specifies the domain directory and the `sca>HelloWorld.jar` SAR file.

```
wls:/mydomain/ServerConfig> sca_unregisterCompositeOfflineDeployment
('/scratch/aimel/fmwhome12/user_projects/domains/soainfra','/tmp/sca>HelloWorld.jar')
```

This example specifies the domain directory and the `shareddata.jar` shared data file.

```
wls:/mydomain/ServerConfig> sca_unregisterCompositeOfflineDeployment('/scratch/aimel/
fmwhome12/
user_projects/domains/soainfra','/tmp/shareddata.jar')
```

1.14 SOA Health Check Commands

The SOA Health Check framework provides a number of health checks and health check categories to monitor the health of your system. Checks include memory checks, data source checks, SOA-related service checks, SOA application-targeting checks, BPEL-related checks, and so on.

Use the SOA Health Check commands, listed in [Table 1-14](#), to configure and execute SOA health checks.

Table 1-14 SOA Health Check Commands

Use this command...	To...	Use with WLST...
executeHealthCheckAndGenerateReport	Execute a health check and retrieve the report.	Online
executeHealthCheckCategoryAndGenerateReport	Execute a health check category and retrieve the report.	Online
executeHealthChecksAndGenerateReport	Execute the specified health checks and retrieve the report.	Online
getUnsuccessfulHCRequestIds	Retrieve all unsuccessful run ids in the specified time interval.	Online
getUnsuccessfulHCRequestResultReport	Retrieve the report corresponding to the specified unsuccessful health check run-id.	Online
listHealthChecks	List all registered health checks.	Online
listHealthChecksInCategory	List all health checks in the specified category.	Online
listHealthCheckCategories	List all health check categories.	Online
deleteAllHCRequestResults	Delete all health check request results.	Online

Table 1-14 (Cont.) SOA Health Check Commands

Use this command...	To...	Use with WLST...
deleteHCRequestResults	Delete health check request results gathered during the specified time interval.	Online
getUnsuccessfulHCResultReport	Retrieve the results report of unsuccessful health check runs that match a filter criteria.	Online
getHCPParameter	Retrieve the current value of a health check parameter.	Online
setHCPParameter	Set the value for a health check parameter.	Online
resetHCPParameter	Reset a health check parameter to its default value.	Online
getHCPParameters	Retrieve the list of health check parameters and their values.	Online
enableHCStartupRun	Enable or disable the health check startup run.	Online
setHCStartupRunCategory	Change the category used for the health check startup run.	Online
getHCStartupRun	Retrieve the health check startup run details.	Online
createHCPeriodicRun	Create a health check periodic run with the specified name and settings.	Online
enableHCPeriodicRun	Enable or disable a health check periodic run.	Online
setHCPeriodicRunInterval	Change the interval for a periodic run.	Online
setHCPeriodicRunSchedule	Change the schedule for a periodic run.	Online
setHCPeriodicRunCategory	Change the category of a periodic run to the specified category.	Online
getHCPeriodicRun	Retrieve settings for the specified periodic run.	Online
getHCPeriodicRuns	Retrieve details of all health check periodic runs.	Online
removeHCPeriodicRun	Delete a health check periodic run.	Online
getDisabledHealthChecks	Retrieve the list of disabled health checks.	Online
addDisabledHealthCheck	Disable a health check.	Online
removeDisabledHealthCheck	Re-enable a disabled health check.	Online

1.14.1 executeHealthCheckAndGenerateReport

Command Category: SOA Health Check Commands

Use with WLST: Online

Description

Executes the specified health check and retrieves a report of the run.

Syntax

```
executeHealthCheckAndGenerateReport(hcName, fileName)
```

Argument	Definition
<i>hcName</i>	Name of the health check to execute.
<i>fileName</i>	Name and path of the file used to store the output report.

Example 1-1 Example

The following example executes the DatasourceCheck health check and stores the result in /tmp/myfile.

```
wls:/mydomain/ServerConfig>  
executeHealthCheckAndGenerateReport('DatasourceCheck',  
'/tmp/myfile')
```

1.14.2 executeHealthCheckCategoryAndGenerateReport

Command Category: SOA Health Check Commands

Use with WLST: Online

Description

Executes the specified health check category and retrieves a report of the run.

Syntax

```
executeHealthCheckCategoryAndGenerateReport(categoryName, fileName)
```

Argument	Definition
<i>categoryName</i>	Name of the health check category to execute.
<i>fileName</i>	Name and path of the file used to store the output report.

Example 1-2 Example

The following example executes the Startup health check category and stores the result in /tmp/myfile.

```
wls:/mydomain/ServerConfig>
executeHealthCheckCategoryAndGenerateReport('Startup',
'/tmp/myfile')
```

1.14.3 executeHealthChecksAndGenerateReport

Command Category: SOA Health Check Commands

Use with WLST: Online

Description

Executes the specified health checks and retrieves a report of the run.

Syntax

```
executeHealthChecksAndGenerateReport(hcNames, fileName)
```

Argument	Definition
<i>hcNames</i>	Names of the health checks to execute.
<i>fileName</i>	Name and path of the file used to store the output report.

Example 1-3 Example

```
wls:/mydomain/ServerConfig>
executeHealthChecksAndGenerateReport(['DataSourceCheck',
'SOALibraryCheck'],
'/tmp/myfile')
```

1.14.4 getUnsuccessfulHCRequestIds

Command Category: SOA Health Check Commands

Use with WLST: Online

Description

Retrieves all unsuccessful run ids in the specified time interval.

Syntax

```
getUnsuccessfulHCRequestIds(startTime, endTime)
```

Argument	Definition
<i>startTime</i>	Start time of the interval.
<i>endTime</i>	End time of the interval.

Example 1-4 Example

The following example retrieves all unsuccessful run ids between 1:00 pm and 3:00 pm on 17th June, 2014.

```
wls:/mydomain/ServerConfig> getUnsuccessfulHCRequestIds('Jun 17, 2014  
1:00:00 PM',  
'Jun 17, 2014 3:00:00 PM')
```

1.14.5 getUnsuccessfulHCRequestResultReport

Command Category: SOA Health Check Commands

Use with WLST: Online

Description

Retrieves the report corresponding to the specified unsuccessful health check run-id.

Syntax

```
getUnsuccessfulHCRequestResultReport(id, fileName)
```

Argument	Definition
<i>id</i>	Health check run id.
<i>fileName</i>	Name and path of the file used to store the output report.

Example 1-5 Example

The following example retrieves the report corresponding to run id 2 and stores it in /tmp/myfile.

```
wls:/mydomain/ServerConfig> getUnsuccessfulHCRequestResultReport('2',  
'/tmp/myfile')
```

1.14.6 listHealthChecks

Command Category: SOA Health Check Commands

Use with WLST: Online

Description

Lists all registered health checks.

Syntax

```
listHealthChecks()
```


Example 1-6 Example

The following example lists the registered health checks.

```
wls:/mydomain/ServerConfig> listHealthChecks()

Successfully executed the command. Registered health checks are:
-----
[DatasourceCheck]
[SOAppCheck]
[JDBCpoolsCheck]
[SOALibraryCheck]
[BPELAsyncRequestCheck]
[BPELComponentInstanceCheck]
[EDNMessageCheck]
[ServiceCheck]
[MemoryCheck]
-----
```

1.14.7 listHealthChecksInCategory

Command Category: SOA Health Check Commands

Use with WLST: Online

Description

Lists all health checks in the specified category.

Syntax

```
listHealthChecksInCategory(categoryName)
```

Argument	Definition
<i>categoryName</i>	Name of the health check category.

Example 1-7 Example

The following example lists all health checks under the Startup category.

```
wls:/mydomain/ServerConfig> listHealthChecksInCategory('Startup')

Successfully executed the command. Registered health checks in the given
category are:
-----
-----
---
---
[DatasourceCheck]
[SOAppCheck]
[JDBCpoolsCheck]
[SOALibraryCheck]
[ServiceCheck]
[MemoryCheck]
-----
```


1.14.8 listHealthCheckCategories

Command Category: SOA Health Check Commands

Use with WLST: Online

Description

Lists all health check categories.

Syntax

```
listHealthCheckCategories()
```

Example 1-8 Example

The following example lists the health check categories:

```
wls:/mydomain/ServerConfig> listHealthCheckCategories()

Successfully executed the command. Registered health check categories
are:
-----
--
[Startup]
[Quick]
[Intermediate]
[Extensive]
[Config]
-----
--
```

1.14.9 deleteAllHCRequestResults

Command Category: SOA Health Check Commands

Use with WLST: Online

Description

Deletes all health check request results.

Syntax

```
deleteAllHCRequestResults()
```

Example 1-9 Example

The following example deletes all health check request results:

```
wls:/mydomain/ServerConfig> deleteAllHCRequestResults()

Successfully executed the command.
```

1.14.10 deleteHCRequestResults

Command Category: SOA Health Check Commands

Use with WLST: Online

Description

Deletes health check request results gathered during the specified time interval.

Syntax

```
deleteHCRequestResults(startTime, endTime)
```

Argument	Definition
<i>startTime</i>	Start time of the interval.
<i>endTime</i>	End time of the interval.

Example 1-10 Example

The following example deletes health check request results between 1:00 pm and 3:00 pm on 17th June, 2014.

```
wls:/mydomain/ServerConfig> deleteHCRequestResults('Jun 17, 2014 1:00:00
PM',
'Jun 17, 2014 3:00:00 PM')
```

1.14.11 getUnsuccessfulHCResultReport

Command Category: SOA Health Check Commands

Use with WLST: Online

Description

Retrieves the results report of unsuccessful health check runs that match the specified filter criteria.

If *stateNames* is not passed, then all states are used. If *hcName* is not passed, then all health checks are considered. If *fileName* is not passed, a report called *hcResultReport.html* is generated in the current working directory.

Syntax

```
getUnsuccessfulHCResultReport(startTime, endTime, stateNames=[],
hcName=None, fileName='hcResultReport.html')
```

Argument	Definition
<i>startTime</i>	Start time of the interval.
<i>endTime</i>	End time of the interval.
<i>stateNames</i>	Optional. State names (for example, ['Success'])

Argument	Definition
<i>hcName</i>	Optional. Name of the health check for which unsuccessful result reports are to be retrieved.
<i>fileName</i>	Optional. Name and path of the file in which the output report is to be stored.

Example 1-11 Example

```
wls:/mydomain/ServerConfig> getUnsuccessfulHCResultReport('Jan 1, 1970
0:00:00
AM', 'Jul 22, 2014 8:00:00 PM', ['Success'], 'DatasourceCheck',
'/tmp/myfile')
```

1.14.12 getHCPParameter

Command Category: SOA Health Check Commands

Use with WLST: Online

Description

Retrieves the current value of the specified health check parameter.

Syntax

```
getHCPParameter(paramName)
```

Argument	Definition
<i>paramName</i>	Name of the health check parameter to check.

Example 1-12 Example

```
wls:/mydomain/ServerConfig> getHCPParameter("MemoryCheck.memThreshold")

Health Check Parameter:
-----
Name : MemoryCheck.memThreshold
Value : 25
```

1.14.13 setHCPParameter

Command Category: SOA Health Check Commands

Use with WLST: Online

Description

Sets the specified value for a health check parameter. This command overrides the default value of the health check parameter.

Syntax

```
setHCPParameter(paramName, paramValue)
```

Argument	Definition
<i>paramName</i>	Name of the health check parameter to update.
<i>paramValue</i>	New value of the health check parameter.

Example 1-13 Example

```
wls:/mydomain/ServerConfig> setHCPParameter("MemoryCheck.memThreshold",  
"35")
```

1.14.14 resetHCPParameter

Command Category: SOA Health Check Commands

Use with WLST: Online

Description

Resets the specified health check parameter to its default value.

Syntax

```
resetHCPParameter(paramName)
```

Argument	Definition
<i>paramName</i>	Name of the health check parameter to update.

Example 1-14 Example

```
wls:/mydomain/ServerConfig> resetHCPParameter("MemoryCheck.memThreshold")
```

1.14.15 getHCPParameters

Command Category: SOA Health Check Commands

Use with WLST: Online

Description

Retrieves the list of health check parameters and their values.

Syntax

```
getHCPParameters()
```

Example 1-15 Example

```
wls:/mydomain/ServerConfig> getHCPParameters()
```

```
Successfully executed the command. Health check parameters are:
-----
JDBC Pools Check.ConnectionMaxMillis=2000
EDNMessageCheck.pendingMsgOaooCount=5000
EDNMessageCheck.pendingMsgCount=2000
BPELAsyncRequestCheck.pendingAsyncReqCount=1000
BPELComponentInstanceCheck.faultInstanceCount=500
BPELComponentInstanceCheck.openInstanceCount=1000
JDBC Pools Check.TestMaxMillis=1000
MemoryCheck.memThreshold=35
-----
```

1.14.16 enableHCStartupRun

Command Category: SOA Health Check Commands

Use with WLST: Online

Description

Enables or disables the health check startup run.

Syntax

```
enableHCStartupRun(enabledFlag)
```

Argument	Definition
<i>enabledFlag</i>	Boolean value (true or false) used to enable or disable the health check startup run.

Example 1-16 Example

The following example enables the startup run.

```
wls:/mydomain/ServerConfig> enableHCStartupRun(true)
```

1.14.17 setHCStartupRunCategory

Command Category: SOA Health Check Commands

Use with WLST: Online

Description

Changes the category used for the health check startup run.

Syntax

```
setHCStartupRunCategory(categoryName)
```

Argument	Definition
<i>categoryName</i>	Name of the health check category to be used for the startup run.

Example 1-17 Example

The following example sets the startup run category to Quick.

```
wls:/mydomain/ServerConfig> setHCStartupRunCategory('Quick')
```

1.14.18 getHCStartupRun

Command Category: SOA Health Check Commands

Use with WLST: Online

Description

Retrieves the health check startup run details. The details include the startup run health check category and the status of the run (enabled or disabled).

Syntax

```
getHCStartupRun()
```

Example 1-18 Example

```
wls:/mydomain/ServerConfig> getHCStartupRun()
```

```
Startup Run
-----
Category Name : Startup
Enabled       : true
```

1.14.19 createHCPeriodicRun

Command Category: SOA Health Check Commands

Use with WLST: Online

Description

Creates a health check periodic run with the specified name and settings.

Syntax

```
createHCPeriodicRun(name, enabledFlag, interval, minuteOfHour, hourOfDay,
dayOfWeek, categoryName)
```

Argument	Definition
<i>name</i>	Name of the periodic run.
<i>enabledFlag</i>	Boolean value (true or false) that determines whether the periodic run is enabled or disabled.
<i>interval</i>	Interval or time period (in minutes) between two periodic runs.
<i>minuteOfHour</i>	Minute of the hour at which to start the first run. The value -1 stands for immediate.

Argument	Definition
<i>hourOfDay</i>	Hour of the day at which to start the first run. The value -1 stands for immediate.
<i>dayOfWeek</i>	Day of the week at which to start the first run. The value -1 stands for immediate.
<i>categoryName</i>	Name of the health check category to be used for the periodic run.

Example 1-19 Example

```
wls:/mydomain/ServerConfig> createHCPeriodicRun('myrun', true, 30, 0, -1,
-1,
'Intermediate')
```

1.14.20 enableHCPeriodicRun

Command Category: SOA Health Check Commands

Use with WLST: Online

Description

Enables or disables a health check periodic run.

Syntax

```
enableHCPeriodicRun(name, enabledFlag)
```

Argument	Definition
<i>name</i>	Name of the periodic run to enable or disable.
<i>enabledFlag</i>	Boolean value (true or false) that determines whether the periodic run is to be enabled or disabled.

Example 1-20 Example

The following example enables the myrun periodic health check:

```
wls:/mydomain/ServerConfig> enableHCPeriodicRun('myrun', true)
```

1.14.21 setHCPeriodicRunInterval

Command Category: SOA Health Check Commands

Use with WLST: Online

Description

Changes the interval of the specified periodic run.

Syntax

```
setHCPeriodicRunInterval(name, interval)
```


Argument	Definition
<i>name</i>	Name of the health check periodic run.
<i>interval</i>	Time interval, in minutes, between successive executions of the specified periodic run.

Example 1-21 Example

The following example sets the interval to 2 hours for the myrun periodic health check:

```
wls:/mydomain/ServerConfig> setHCPeriodicRunInterval('myrun', 120)
```

1.14.22 setHCPeriodicRunSchedule

Command Category: SOA Health Check Commands

Use with WLST: Online

Description

Changes the schedule for the specified periodic run.

Syntax

```
setHCPeriodicRunSchedule(name, minuteOfHour, hourOfDay, dayOfWeek)
```

Argument	Definition
<i>name</i>	Name of the health check periodic run.
<i>minuteOfHour</i>	Minute of the hour at which to start the first run. The value -1 stands for immediate.
<i>hourOfDay</i>	Hour of the day at which to start the first run. The value -1 stands for immediate.
<i>dayOfWeek</i>	Day of the week at which to start the first run. The value -1 stands for immediate.

Example 1-22 Example

```
wls:/mydomain/ServerConfig> setHCPeriodicRunSchedule('hourly', 0, -1, -1)
```

1.14.23 setHCPeriodicRunCategory

Command Category: SOA Health Check Commands

Use with WLST: Online

Description

Changes the category of a periodic run to the specified category.

Syntax

```
setHCPeriodicRunCategory(name, categoryName)
```

Argument	Definition
<i>name</i>	Name of the health check periodic run.
<i>categoryName</i>	Name of the health check category to be used for the periodic run.

Example 1-23 Example

```
wls:/mydomain/ServerConfig> setHCPeriodicRunCategory('myrun', 'Extensive')
```

1.14.24 getHCPeriodicRun

Command Category: SOA Health Check Commands

Use with WLST: Online

Description

Retrieves details (like category, schedule, and enabled status) for the specified periodic run.

Syntax

```
getHCPeriodicRun(name)
```

Argument	Definition
<i>name</i>	Name of the health check periodic run.

Example 1-24 Example

```
wls:/mydomain/ServerConfig> getHCPeriodicRun('myrun')
```

```
Periodic Run
-----
Name           : myrun
Category Name  : Extensive
Enabled        : false
Interval       : 2
Day of week    : -1
Hour of day    : -1
Minute of hour : 0
```

1.14.25 getHCPeriodicRuns

Command Category: SOA Health Check Commands

Use with WLST: Online

Description

Retrieves the details of all the health check periodic runs.

Syntax

```
getHCPeriodicRuns()
```

Example 1-25 Example

```
wls:/mydomain/ServerConfig> getHCPeriodicRuns()
```

```
Periodic Run Table
```

```
-----
```

```
PeriodicRun: frequently, enabled=false, interval=60, minuteOfHour=7,
hourOfDay=-1,
dayOfWeek=-1, category=Quick
PeriodicRun: daily, enabled=false, interval=1440, minuteOfHour=15,
hourOfDay=5,
dayOfWeek=-1, category=Intermediate
PeriodicRun: weekly, enabled=false, interval=10080, minuteOfHour=30,
hourOfDay=3,
dayOfWeek=1, category=Extensive
PeriodicRun: myrun, enabled=false, interval=2, minuteOfHour=0,
hourOfDay=-1,
dayOfWeek=-1, category=Extensive
PeriodicRun: myrun2, enabled=false, interval=2, minuteOfHour=0,
hourOfDay=-1,
dayOfWeek=-1, category=Intermediate
PeriodicRun: myrun3, enabled=false, interval=2, minuteOfHour=0,
hourOfDay=-1,
dayOfWeek=-1, category=Intermediate
PeriodicRun: myrun4, enabled=false, interval=30, minuteOfHour=0,
hourOfDay=-1,
dayOfWeek=-1, category=Intermediate
```

1.14.26 removeHCPeriodicRun

Command Category: SOA Health Check Commands

Use with WLST: Online

Description

Deletes the specified health check periodic run.

Syntax

```
removeHCPeriodicRun(name)
```

Argument	Definition
<i>name</i>	Name of the health check periodic run to delete.

Example 1-26 Example

```
wls:/mydomain/ServerConfig> removeHCPeriodicRun('myrun')
```

1.14.27 getDisabledHealthChecks

Command Category: SOA Health Check Commands

Use with WLST: Online

Description

Retrieves the list of disabled health checks.

Syntax

```
getDisabledHealthChecks()
```

Example 1-27 Example

```
wls:/mydomain/ServerConfig> getDisabledHealthChecks()
```

1.14.28 addDisabledHealthCheck

Command Category: SOA Health Check Commands

Use with WLST: Online

Description

Disables the specified health check.

Syntax

```
addDisabledHealthCheck(hcName)
```

Argument	Definition
<i>hcName</i>	Name of the health check to disable.

Example 1-28 Example

```
wls:/mydomain/ServerConfig> addDisabledHealthCheck('DatasourceCheck')
```

1.14.29 removeDisabledHealthCheck

Command Category: SOA Health Check Commands

Use with WLST: Online

Description

Re-enables the disabled health check.

Syntax

```
removeDisabledHealthCheck(hcName)
```

Argument	Definition
<i>hcName</i>	Name of the health check to enable.

Example 1-29 Example

```
wls:/mydomain/ServerConfig> removeDisabledHealthCheck('DatasourceCheck')
```

2

Oracle Business Process Management Custom WLST Commands

This chapter lists and describes the custom WLST commands for Oracle Business Process Management.

2.1 BPMLifecycleAdmin Command Group

Table 2-1 lists and describes the BPMLifecycleAdmin commands for project lifecycle administration.

Table 2-1 BPMLifecycleAdmin Commands for Project Lifecycle Administration

Use this command...	To...	Use with WLST...
create_public_share	Create a public share	Offline
unlock_public_share	Unlock a public share	Offline
export_public_share	Export a public share to the file system	Offline
delete_public_share	Delete a public share	Offline
publish_template	Publish a template to MDS	Offline
export_template	Export a template to the file system	Offline
delete_template	Delete a template from MDS	Offline

2.1.1 create_public_share

Command Category: BPMLifecycleAdmin Commands

Use with WLST: Offline

2.1.1.1 Description

Use this command to create a public share from a template. The template must exist in MDS.

2.1.1.2 Syntax

```
create_public_share(composerUser, composerPassword, connectionURL, templateName,  
publicshareId, mdsconfigLocation, [Override], [oracleHome] )
```

Argument	Definition
<i>composerUser</i>	The Oracle Business Process Composer user who performs the current operation.

Argument	Definition
<i>composerPassword</i>	BPM Composer user's password
<i>connectionURL</i>	JNDI connection URL to the security server service in format <i>host:port</i>
<i>templateName</i>	Name of the template in MDS
<i>publicshareId</i>	Name of the public share to be created
<i>mdsconfigLocation</i>	Location of the <i>mds-config.xml</i> to be used to connect to MDS
<i>projectLocation</i>	The path where the public share will be created. If the path does not exist it will be created. The root is '/'.
<i>Override</i>	Enables you to override the public share if a public share exists in MDS with the same name. The template is not overwritten when you execute this command.
<i>oracleHome</i>	Optional. The Oracle home to be used.

2.1.1.3 Examples

The following example creates a public share named `Sample_PublicShare`. It is based on the template with name `Sample_Template`. The name of the public share is `Sample_PublicShare`, and the location of the `mds-config.xml` file is `/tmp/mds-config.xml`.

```
create_public_share('user_name', 'password', 'host:port', 'Sample_Template',
'Sample_PublicShare', '/tmp/mds-config.xml')
```

The following example creates a public share named `Sample_PublicShare`. It is based on the template named `Sample_Template` that exists in MDS. The public share, not the template, is overridden. The location of the `mds-config.xml` file is `/tmp/mds-config.xml`.

```
create_public_share('user_name', 'password', 'host:port', 'Sample_Template',
'Sample_PublicShare', '/tmp/mds-config.xml', 'true')
```

2.1.2 unlock_public_share

Command Category: BPMLifecycleAdmin Commands

Use with WLST: Offline

2.1.2.1 Description

Use this command to unlock a public share. For example, when you create project by using the Ant task `create_public_share` command, the project is created as locked. You can then unlock it by using the `unlock_public_share` command.

A lock is also set by enabling or disabling the check box **enable sharing** in the project creation page in Oracle Business Process Composer.

It is also released when the user publishes a project from Oracle Business Process Composer.

The public share must exist in MDS.

2.1.2.2 Syntax

```
unlock_public_share(composerUser, composerPassword, connectionURL, publicshareId,
mdsconfigLocation, [oracleHome] )
```

Argument	Definition
<i>composerUser</i>	The Oracle Business Process Composer user who performs the current operation.
<i>composerPassword</i>	BPM Composer user's password
<i>connectionURL</i>	JNDI connection URL to the security server service in format <i>host:port</i>
<i>publicshareId</i>	Name of the public share to be unlocked
<i>mdsconfigLocation</i>	Location of the <i>mds-config.xml</i> to be used to connect to MDS
<i>oracleHome</i>	Optional. The Oracle home to be used

2.1.2.3 Example

The following example unlocks a public share named *Sample_PublicShare*. The location of the *mds-config.xml* file is */tmp/mds-config.xml*.

```
unlock_public_share('user_name', 'password', 'host:port', 'Sample_PublicShare', '/tmp/
mds-config.xml')
```

2.1.3 export_public_share

Command Category: BPMLifecycleAdmin Commands

Use with WLST: Offline

2.1.3.1 Description

Use this command to export the public share from MDS to the file system.

2.1.3.2 Syntax

```
export_public_share(composerUser, composerPassword, connectionURL,
publicshareId,fsLocation, mdsconfigLocation, [oracleHome] )
```

Argument	Definition
<i>composerUser</i>	The Oracle Business Process Composer user who performs the current operation.

Argument	Definition
<i>composerPassword</i>	BPM Composer user's password
<i>connectionURL</i>	JNDI connection URL to the security server service in format <i>host:port</i>
<i>publicshareId</i>	Name of the public share to be exported
<i>fsLocation</i>	File system location where the project is to be downloaded
<i>mdsconfigLocation</i>	Location of the <i>mds-config.xml</i> to be used to connect to MDS
<i>oracleHome</i>	Optional. The Oracle home to be used

2.1.3.3 Example

The following example specifies the public share name as `Sample_PublicShare`, the file system location as `/tmp`, and the location of the `mds-config.xml` file as `/tmp/mds-config.xml`.

```
export_public_share('user_name', 'password', 'host:port', 'Sample_PublicShare', '/tmp', '/tmp/mds-config.xml')
```

2.1.4 delete_public_share

Command Category: BPMLifecycleAdmin Commands

Use with WLST: Offline

2.1.4.1 Description

Use this command to delete a public share from MDS. Executing this command requires that the public share is not locked.

2.1.4.2 Syntax

```
delete_public_share(composerUser, composerPassword, connectionURL, publicshareId, mdsconfigLocation, [releaseLock], [oracleHome] )
```

Argument	Definition
<i>composerUser</i>	The Oracle Business Process Composer user who performs the current operation.
<i>composerPassword</i>	BPM Composer user's password
<i>connectionURL</i>	JNDI connection URL to the security server service in format <i>host:port</i>
<i>publicshareId</i>	Name of the public share to be deleted

Argument	Definition
<i>mdsconfigLocation</i>	Location of the <code>mds-config.xml</code> to be used to connect to MDS
<i>releaseLock</i>	Optional. If the public share is locked, this lock can be released and the delete operation completed. You can set this attribute to either <code>true</code> or <code>false</code> . If not specified, default value is <code>false</code> .
<i>oracleHome</i>	Optional. The Oracle home to be used

2.1.4.3 Examples

The following example specifies the name and location of a public share to be deleted.

```
delete_public_share('Sample_PublicShare','/tmp/mds-config.xml')
```

The following example specifies the name and location of a public share to be deleted, and that the public share should be deleted even if locked.

```
delete_public_share('user_name', 'password', 'host:port','Sample_PublicShare','/tmp/mds-config.xml','true')
```

2.1.5 publish_template

Command Category: BPMLifecycleAdmin Commands

Use with WLST: Offline

2.1.5.1 Description

Use this command to publish the template from the file system to MDS.

2.1.5.2 Syntax

```
publish_template(composerUser, composerPassword, connectionURL,
templateName,fsLocation, mdsconfigLocation, [Override], [oracleHome] )
```

Argument	Definition
<i>composerUser</i>	The Oracle Business Process Composer user who performs the current operation.
<i>composerPassword</i>	BPM Composer user's password
<i>connectionURL</i>	JNDI connection URL to the security server service in format <code>host:port</code>
<i>templateName</i>	Name of the template to be published
<i>fsLocation</i>	File system location of the template project
<i>mdsconfigLocation</i>	Location of the <code>mds-config.xml</code> to be used to connect to MDS

Argument	Definition
<code>projectLocation</code>	The path where the public share will be created. If the path does not exist it will be created. The root is '/' .
<code>Override</code>	When you publish a template in MDS, this attribute enables you to override an existing template with the same name. Can either be 'true' or 'false'. If not specified, default value is 'false'.
<code>oracleHome</code>	Optional. The Oracle home to be used

2.1.5.3 Example

The following example publishes a template named `Sample_Template_Name_MDS`. to the root folder.

```
f('user_name', 'password', 'host:port', 'Sample_Template', '/tmp/MyTemplate', '/',
'/tmp/mds-config.xml')
```

The following example publishes a template named `Sample_Template_Name_MDS`.to the `WorkingOn/` folder.

```
publish_template('user_name', 'password', 'host:port', 'Sample_Template', '/tmp/
MyTemplate', 'WorkingOn', '/tmp/mds-config.xml')
```

2.1.6 export_template

Command Category: BPMLifecycleAdmin Commands

Use with WLST: Offline

2.1.6.1 Description

Use this command to export the template from MDS to the file system.

2.1.6.2 Syntax

```
export_template(composerUser, composerPassword, connectionURL, templateName,
fsLocation, mdsconfigLocation, [oracleHome] )
```

Argument	Definition
<code>composerUser</code>	The Oracle Business Process Composer user who performs the current operation.
<code>composerPassword</code>	BPM Composer user's password
<code>connectionURL</code>	JNDI connection URL to the security server service in format <code>host:port</code>
<code>templateName</code>	Name of the template to be exported
<code>fsLocation</code>	File system location where the project is to be downloaded

Argument	Definition
<i>mdsconfigLocation</i>	Location of the <code>mds-config.xml</code> to be used to connect to MDS
<i>oracleHome</i>	Optional. The Oracle home to be used

2.1.6.3 Example

The following example specifies the template name as `Sample_Template`, the file system location as `/tmp`, and the location of the `mds-config.xml` file as `/tmp/mds-config.xml`.

```
export_template('user_name', 'password', 'host:port', 'Sample_Template', '/tmp', '/tmp/mds-config.xml')
```

2.1.7 delete_template

Command Category: BPMLifecycleAdmin Commands

Use with WLST: Offline

2.1.7.1 Description

Use this command to delete the template from MDS.

2.1.7.2 Syntax

```
delete_template(composerUser, composerPassword, connectionURL,
templateName, mdsconfigLocation, [oracleHome] )
```

Argument	Definition
<i>composerUser</i>	The Oracle Business Process Composer user who performs the current operation.
<i>composerPassword</i>	BPM Composer user's password
<i>connectionURL</i>	JNDI connection URL to the security server service in format <code>host:port</code>
<i>templateName</i>	Name of the template to be deleted
<i>fsLocation</i>	File system location of the template project
<i>mdsconfigLocation</i>	Location of the <code>mds-config.xml</code> to be used to connect to MDS
<i>projectLocation</i>	The path where the public share will be created. If the path does not exist it will be created. The root is <code>'/'</code> .
<i>oracleHome</i>	Optional. The Oracle home to be used

2.1.7.3 Example

The following example deletes the template named `Sample_template` from MDS.

```
delete_template('weblogic', 'welcome1', 'host:port', '/Sample_template', '/tmp/mds-  
config.xml')
```

3

Oracle Enterprise Scheduler Custom WLST Commands

Use the Oracle Enterprise Scheduler (ESS) commands to manage Oracle Enterprise Scheduler jobs. Many of the commands have analogous convenience shell scripts to make execution simpler. The native commands are described in [Oracle Enterprise Scheduler Custom Native Commands](#) and the shell scripts are described in [Oracle Enterprise Scheduler Convenience Scripts](#).

Note:

To use these Oracle Enterprise Scheduler custom WLST commands, you must invoke the WLST script from the Oracle Common home. See "Using Custom WLST Commands" in the *Administering Oracle Fusion Middleware*.

This chapter includes the following sections:

- [Native Invocation](#)
- [Oracle Enterprise Scheduler Custom Native Commands](#)
- [Oracle Enterprise Scheduler Batch Delete Requests](#)
- [Work Allocation Commands](#)
- [Oracle Enterprise Scheduler Diagnostic Dumps](#)
- [Oracle Enterprise Scheduler Convenience Scripts](#)

3.1 Native Invocation

The following steps describe how to run the native WLST commands described in this chapter. Convenience shell scripts are available for many of these commands and are described in [Oracle Enterprise Scheduler Convenience Scripts](#).

1. Change directory to:

```
$ORACLE_HOME/common/bin
```

2. Place the following JAR archives in your class path:

- `MW_HOME/oracle_common/modules/oracle.jmx_11.1.1/jmxframework.jar`
- `MW_HOME/WLS_HOME/lib/weblogic.jar`
- `MW_HOME/ORACLE_HOME/ess/lib/ess-admin.jar`

3. Execute one of the following commands:

- `wlst.sh` (Linux)
- `wlst.cmd` (Windows)

4. Connect to the WLS Admin Server. For example:

```
connect('weblogic', 'weblogic1', 't3://hostabc:7001')
```

5. Invoke the WLST commands.

3.1.1 Enabling and Disabling Print Statements

You can explicitly enable or disable Oracle Enterprise Scheduler print output as shown below. For interactive use, you must enable printing. By default, Oracle Enterprise Scheduler printing is enabled.

- `enableESSPrint()` — Enables print output for Oracle Enterprise Scheduler WLST commands.
- `disableESSPrint()` — Disables print output for Oracle Enterprise Scheduler WLST commands.

3.1.2 Getting Help

While in the WLST console, you can get the list of available Oracle Enterprise Scheduler native commands by typing:

```
wls:/offline> help('OracleScheduler')
```

For usage information about any command, type `help('command_name')`. For example:

```
wls:/offline> help('manageSchedulerRuntimeConfig')
```

3.2 Oracle Enterprise Scheduler Custom Native Commands

Use the Oracle Enterprise Scheduler commands listed in [Table 3-1](#) to manage the Oracle Enterprise Scheduler server, configuration, job requests, and logs. In the third 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.

Table 3-1 Oracle Enterprise Scheduling Service Basic Commands

Use this command...	To...	Use with WLST...
<code>manageSchedulerRuntimeConfig</code>	Add, modify, delete and display various configuration parameters.	Online
<code>getSchedulerRequestContent</code>	Get the log and output data files for a request after its execution is completed.	Online
<code>querySchedulerRequests</code>	Search and list requests based upon hosting application name, state or elapsed time of execution.	Online
<code>manageSchedulerRequest</code>	Manually cancel, recover, or complete request state.	Online
<code>manageSchedulerServer</code>	Start, stop or get status of the Oracle Enterprise Scheduler application running on the server.	Online

Table 3-1 (Cont.) Oracle Enterprise Scheduling Service Basic Commands

Use this command...	To...	Use with WLST...
<code>submitSchedulerRequest</code>	Submit a job request to Oracle Enterprise Scheduler for execution.	Online
<code>manageSchedulerJobDefn</code>	Manage (show, create, delete, customize, update) an Oracle Enterprise Scheduler job definition.	Online
<code>manageSchedulerSchedule</code>	Manage (show, create, delete, update) a schedule definition in Oracle Enterprise Scheduler.	Online

3.2.1 Manage (Add/Delete/Modify/Get) Configuration Parameters

Command Category: Application Management Commands

Use with WLST: Online

3.2.1.1 Description

Adds, modifies, deletes and displays various configuration parameters.

3.2.1.2 Syntax

```
manageSchedulerRuntimeConfig(app,type,[operation],[name],[val])
```

Argument	Definition
<code>app</code>	The hosting application name for managing run time configuration.
<code>type</code>	The property type. Can be either APP or ESS.
<code>operation</code>	Optional. The operation to perform. One of the following: <ul style="list-style-type: none"> Add (<code>add</code>) Modify (<code>mod</code>) Delete (<code>del</code>) Get (<code>get</code>) Get All (<code>getall</code>)
<code>name</code>	(Optional for the <code>getall</code> operation) The name of the configuration parameter.
<code>val</code>	(Optional for the <code>del/get/getall</code> operations) Value to set for the parameter.

3.2.1.3 Examples

- Add an ENV parameter `foo` with value "bar":

```
manageSchedulerRuntimeConfig("myapp", "APP", operation="add", name="foo", val="bar")
```
- Get the value of ENV parameter `foo`:


```
manageSchedulerRuntimeConfig("myapp", "APP", operation="get", name="foo")
```

- Get the list of all ENV parameters:

```
manageSchedulerRuntimeConfig("myapp", "APP", operation="getall")
manageSchedulerRuntimeConfig("myapp", "APP")
```

- Change the value of the ENV parameter `foo` to "barone":

```
manageSchedulerRuntimeConfig("myapp", "APP", operation="mod", name="foo",
val="barone")
```

- Delete the ENV parameter `foo`:

```
manageSchedulerRuntimeConfig("myapp", "APP", operation="del", name="foo")
```

- Show all parameters of type ESS:

```
manageSchedulerRuntimeConfig("myapp", "ESS")
```

3.2.1.4 Related Shell Script

[essManageRuntimeConfig](#)

3.2.2 Get Log and Output Content of a Request

Command Category: Application Management Commands

Use with WLST: Online

3.2.2.1 Description

Gets the log and output data files for a request after its execution is completed.

3.2.2.2 Syntax

```
getSchedulerRequestContent(requestId,contentType,[logLines], [outDir])
```

Argument	Definition
<i>requestId</i>	request ID
<i>contentType</i>	Type of content to handle. Can be one of the following; <ul style="list-style-type: none"> • LOG • OUTPUT • BINARY_OUTPUT • TEXT_OUTPUT By default, the command checks for both BINARY_OUTPUT and TEXT_OUTPUT.
<i>logLines</i>	Optional. The number of lines to be read from the request log. The default value is 1000.
<i>outDir</i>	Optional. The absolute path of the directory in which to dump the output files. By default, the command uses the current directory.

3.2.2.3 Examples

- Get the request log for request ID 123:

```
getSchedulerRequestContent(123, "LOG")
```

- Get all the output of request 123:

```
getSchedulerRequestContent(123, "OUTPUT")
```

- Get all the output of request 123 and save it in directory /tmp:

```
getSchedulerRequestContent(123, "OUTPUT", outDir="/tmp")
```

- Get all the text output of request 123 and save it in directory /tmp:

```
getSchedulerRequestContent(123, "TEXT_OUTPUT", outDir="/tmp")
```

- Get all the binary output of request 123 and save it in directory /tmp:

```
getSchedulerRequestContent(123, "BINARY_OUTPUT", outDir="/tmp")
```

- Get first 100 lines of the request log for request id 123:

```
getSchedulerRequestContent(123, "LOG", logLines=100)
```

3.2.2.4 Related Shell Script

[essGetOutputContent](#)

3.2.3 Search and List Requests

Command Category: Application Management Commands

Use with WLST: Online

3.2.3.1 Description

Searches and lists requests based upon hosting application name, state or elapsed time of execution. This command can be used to find long running requests.

3.2.3.2 Syntax

```
querySchedulerRequests([app],[state],[days],[hours],[minutes])
```

Argument	Definition
<i>app</i>	Optional. The name of the hosting application

Argument	Definition
<i>state</i>	Optional. Specifies the request state. Can be one of the following: <ul style="list-style-type: none"> • WAIT • READY • RUNNING • COMPLETED • BLOCKED • HOLD • CANCELLING • EXPIRED • CANCELLED • ERROR • WARNING • SUCCEDED • PAUSED • PENDING_VALIDATION • VALIDATION_FAILED • SCHEDULE_ENDED • FINISHED • ERROR_AUTO_RETRY • ERROR_MANUAL_RECOVERY By default, the command specifies RUNNING.
<i>days</i>	Optional. Specifies time in days
<i>hours</i>	Optional. Specifies time in hours
<i>minutes</i>	Optional. Specifies time in hours

3.2.3.3 Examples

- Get all the requests that are in the RUNNING state:

```
querySchedulerRequests()
or
querySchedulerRequests(state="RUNNING")
```
- Get all cancelled requests:

```
querySchedulerRequests(state="CANCELLED")
```
- Get all requests running for more than two days:

```
querySchedulerRequests(days=2)
```
- Get all requests running for more than 10 hours:

```
querySchedulerRequests(hours=10)
```
- Get all requests running for the application myapp:

```
querySchedulerRequests(appName="myapp")
```

3.2.3.4 Shell Script

[essQueryRequests](#)

3.2.4 Manage Requests

Command Category: Application Management Commands

Use with WLST: Online

3.2.4.1 Description

Manually cancel, recover, complete, or get details about requests.

3.2.4.2 Syntax

```
manageSchedulerRequest(requestId,operation,[asyncStatus],[statusMessage])
```

Argument	Definition
<i>requestId</i>	Specifies the request ID
<i>operation</i>	Specifies which one of the following operations to perform: <ul style="list-style-type: none"> • CANCEL • RECOVER • COMPLETE • GETDETAIL
<i>asyncStatus</i>	Required for the COMPLETE operation. Specifies the status to set for the given request. Must be one of the following: <ul style="list-style-type: none"> • SUCCESS • PAUSE • WARNING • ERROR • CANCEL • UPDATE • BIZ_ERROR • ERROR_MANUAL_RECOVERY
<i>statusMessage</i>	Required for the COMPLETE operation. Specifies the qualifying status message to describe the operation

3.2.4.3 Examples

- Cancel request 123:

```
manageSchedulerRequest(123, "CANCEL")
```
- Recover request 123:

```
manageSchedulerRequest(123, "RECOVER")
```
- Complete request 123:

```
manageSchedulerRequest(123, "COMPLETE", asyncStatus="ERROR",  
statusMessage="Completed by Admin")
```

3.2.4.4 Related Shell Script

[essManageRequests](#)

3.2.5 Manage Oracle Enterprise Scheduler Servers

Command Category: Application Management Commands

Use with WLST: Online

3.2.5.1 Description

Starts, stops or gets the status of the Oracle Enterprise Scheduler application running on the server. Starting the Oracle Enterprise Scheduler application starts the Oracle Enterprise Scheduler processor thread so that request processing can start. Stopping Oracle Enterprise Scheduler stops or quiesces the Oracle Enterprise Scheduler processor so that no new requests are processed.

3.2.5.2 Syntax

```
manageSchedulerServer(operation)
```

Argument	Definition
<i>operation</i>	Specifies which one of the following operations to perform: <ul style="list-style-type: none">• START• STOP• STATUS

3.2.5.3 Examples

- Stop Oracle Enterprise Scheduler:

```
manageSchedulerServer("STOP")
```
- Get the current state of the Oracle Enterprise Scheduler processor:

```
manageSchedulerServer("STATUS")
```

3.2.5.4 Related Shell Script

[essManageServer](#)

3.2.6 Submit Job Requests to Oracle Enterprise Scheduler

Command Category: Application Management Commands

Use with WLST: Online

3.2.6.1 Description

Submit a job request to Oracle Enterprise Scheduler for execution.

3.2.6.2 Syntax

```
submitSchedulerRequest(appName,[jobName],[schMeth],[note],[schName],
[RschName],[schDesc],[o_time],[begin_time],[frequency],[freqNum],[count],
[end_time],[month],[week],[day],[date],[reqParams])
```

 **Note:**

While a job definition must be predefined, the schedule can be defined in an ad-hoc manner using this command.

Argument	Definition
<i>appName</i>	Name of the hosting application for the job.
<i>jobName</i>	Name of the job definition to be used for current request. The path name (including the package name) must be fully qualified. If it is not fully qualified, the command uses the default package: <code>/oracle/apps/ess/custom</code> .
<i>schMeth</i>	<p>Optional. Specifies which one of the following scheduling methods to use:</p> <ul style="list-style-type: none"> IMMEDIATE (I) - No other parameter is required. This is the default method and is used if the <i>schMeth</i> argument is not specified. ONCE (O) - If this method is used, the <i>o_time</i> argument is required. SCHEDULE (S) - If this method is used, the <i>schName</i> argument is required. RECURRING (R) - If this method is used, the <i>RschName</i> and <i>frequency</i> arguments are required. <p>Note that the following arguments are only valid with the Recurring(R) method: <i>schDesc</i>, <i>begin_time</i>, <i>freqNum</i>, <i>count</i>, <i>end_time</i>, <i>month</i>, <i>week</i>, <i>day</i>, <i>date</i> and <i>frequency</i>. Parameter combinations based on 'frequency' value are:</p> <p>If you use the recurring (R) method with any of the following options: SECOND MINUTE HOUR DAY WEEK, you must use the following arguments: <i>freqNum</i>, <i>begin_time</i>, [<i>count</i> <i>end_time</i>].</p> <p>If you use the recurring (R) method with the MONTH option, you must use the following arguments [<i>week day</i>] <i>date</i> <i>begin_time</i> [<i>count</i> <i>end_time</i>].</p> <p>If you use the recurring (R) method with the YEAR option, you must use the following arguments: [<i>month</i> ({<i>week day</i>} <i>date</i>)] <i>begin_time</i> [<i>count</i> <i>end_time</i>].</p> <p>Note that when you use the recurring (R) method, the schedule is stored in the MDS for future use and the storage location is based on the schedule name provided.</p> <p>See Examples for examples.</p>
<i>note</i>	Optional. Specifies a short description for the job request.

Argument	Definition
<i>schName</i>	Optional. Specifies the name of the predefined schedule definition. The path name (including the package name) must be fully qualified. If it is not fully qualified, the command uses the default package: <code>/oracle/apps/ess/custom</code> .
<i>RschName</i>	Optional. Specifies the name of the schedule if you use the recurring (R) option to the <i>schMeth</i> argument to define the schedule.
<i>schDesc</i>	Optional. Specifies a string value that describes the ad-hoc schedule. By default, the value is "Custom Ad-Hoc Schedule".
<i>o_time</i>	Optional. Specifies the time for a "one time" execution. The format is: <code>HH:MM:SS:DD:MM:YYYY</code>
<i>begin_time</i>	Optional. Specifies the start time for a recurring schedule. The format is: <code>HH:MM:SS:DD:MM:YYYY</code>
<i>frequency</i>	Optional. Specifies the frequency of a recurring schedule. The valid values are: SECOND, MINUTE, HOUR, DAY, WEEK, MONTH, YEAR
<i>freqNum</i>	Optional. An integer value that specifies repeat interval of a recurring schedule. The default value is "1".
<i>count</i>	Optional. Specifies the maximum number of repetitions for a request.
<i>end_time</i>	Optional. Specifies the end time for a recurring schedule. The format is: <code>HH:MM:SS:DD:MM:YYYY</code>
<i>month</i>	Optional. Specifies the month of the year if <i>frequency</i> is specified as YEARLY. Valid values are: 1-12.
<i>week</i>	Optional. Specifies the week of a month. Valid values are: 1-6, LAST.
<i>day</i>	Optional. Specifies the day of the week. Valid values are: 1-7, where "1" represents Monday.
<i>date</i>	Optional. Specifies a specific day of month. Valid values are: 1-31, LAST
<i>reqParams</i>	Optional. Specifies other request parameters/properties specified as a dictionary.

3.2.6.3 Examples

The following variables are used in these examples:

Variable	Value
HOSTING_APP	EssDemoApp
SCHEDULE_NAME	/oracle/apps/ess/custom/MySch
JOB_DEF_NAME	/oracle/apps/ess/demopackage/BasicJavaJob

- Submit a request for immediate job execution (one-time only):
`submitSchedulerRequest(HOSTING_APP, JOB_DEF_NAME)`
- Submit a request for one-time job execution at a specified time:
`submitSchedulerRequest(HOSTING_APP, JOB_DEF_NAME, schMeth='0', o_time='21:10:30:03:05:2012')`

- Submit a job request with a recurring schedule. The recurrence is: Execution recurs every 2 minutes, starting at time 21:10:30 on 03 May 2012, until 3 iterations complete. The additional request parameters are: *reqName* =test and *PRIORITY*=10. This recurrence also persists in the MDS repository for future use.

```
submitSchedulerRequest(HOSTING_APP, JOB_DEF_NAME, schMeth='R',
RschName=SCHEDULE_NAME, frequency='MINUTE', freqNum=2, count=3,
begin_time='21:10:30:03:05:2012', reqParams={'reqName':'test', 'PRIORITY':'10'})
```

- Submit a job request using a predefined schedule

```
submitSchedulerRequest(HOSTING_APP, JOB_DEF_NAME, schMeth='S',
schName=SCHEDULE_NAME)
```

- Submit a job request with a monthly recurring schedule. The description for the request is "Request with monthly schedule". The recurring schedule name and description are specified using *RschName* and *schDesc* respectively.

```
submitSchedulerRequest(HOSTING_APP, JOB_DEF_NAME, note='Request with monthly
schedule', schMeth='R', RschName=SCHEDULE_NAME+'1', schDesc='Monthly_schedule',
frequency='MONTH', freqNum=2, begin_time='21:10:30:03:05:2012', week=5, day=4,
end_time='21:10:30:04:05:2013')
```

- Submit a job request with a yearly recurring schedule. The description for the request is "Request with yearly schedule". The recurring schedule name and description are provided by *RschName* and *schDesc* respectively.

Execution recurs every three years, on the last day of May (31st), starting at 21:10:30 on 03 May 2012, and iterates five times.

```
submitSchedulerRequest(HOSTING_APP, JOB_DEF_NAME, note='Request with yearly
schedule', schMeth='R', RschName=SCHEDULE_NAME+'2', schDesc='Yearly_schedule',
frequency='YEAR', freqNum=3, begin_time='21:10:30:03:05:2012', month=5,
date='LAST', count=5)
```

- Submit a job request with an hourly recurring schedule. This schedule is stored in the MDS. Recurrence is set to repeat every hour starting from the current time.

```
submitSchedulerRequest(HOSTING_APP, JOB_DEF_NAME, schMeth='R',
RschName=SCHEDULE_NAME, schDesc='HOURLY_schedule', frequency='HOURLY')
```

3.2.6.4 Related Shell Script

[essSubmitRequest](#)

3.2.7 Manage Oracle Enterprise Scheduler Job Definitions

Command Category: Application Management Commands

Use with WLST: Online

3.2.7.1 Description

Manage (show, create, delete, customize, update) an Oracle Enterprise Scheduler job definition.

If *operation* is set to *SHOW* and if *jobName* is specified, the command returns the details of this particular job definition; otherwise, the command shows a list of all job definitions that are part of the application.

If *operation* is set to CREATE, the command returns the `MetadataObjectID` of the newly created job definition.

If *operation* is set to DELETE, CUSTOMIZE or UPDATE, the command returns 0 or -1, depending on whether the operation succeeds or fails.

3.2.7.2 Syntax

```
manageSchedulerJobDefn(operation, appName, [jobName], [jobType], [desc],
[props])
```

Argument	Definition
<i>operation</i>	<p>Specifies which of the following operations to perform on the job:</p> <ul style="list-style-type: none"> • SHOW (S) Displays the list of job definitions that are part of a particular application. If a job name is specified, it shows the details of only that job definition. If this operation is used, the <i>appName</i> argument is required and <i>jobName</i> is optional. • CREATE (N) Creates a new job definition in the namespace of the relevant application (supplied as a parameter) in MDS. If this operation is used, the <i>appName</i>, <i>jobName</i>, and <i>jobType</i> arguments are required and <i>desc</i> and <i>props</i> are optional. • DELETE (D) Deletes the specified job definition from the MDS. If this operation is used, the <i>appName</i> and <i>jobName</i> arguments are required. • CUSTOMIZE (C) Modifies the customizable properties of an existing job definition. The following customizable properties are available: <code>SYS_retries</code>, <code>SYS_priority</code>, <code>SYS_requestCategory</code>, <code>SYS_request_timeout</code>, <code>enableTrace</code>, <code>enableTimeStatistics</code>. If this operation is used, the <i>appName</i>, <i>jobName</i>, and <i>props</i> arguments are required. • UPDATE (U) Updates an existing job definition with the specified property values (existing parameters persist if not overridden). If this operation is used, the <i>appName</i>, <i>jobName</i>, and one or both of the <i>desc</i> and <i>props</i> arguments are required.
<i>appName</i>	Specifies the logical name of the job's hosting application.
<i>jobName</i>	Optional. Specifies the name of the job definition. For the create, delete, or update operations, a default package <code>/oracle/apps/ess/custom/</code> is prepended to the specified name.
<i>jobType</i>	Optional. Specifies the name of the job type to be used to create a new job definition. If the name does not contain the package, the following default value is prefixed to the name: <code>/oracle/as/ess/core/</code> .
<i>desc</i>	Optional. Specifies a description for a new job definition.

Argument	Definition
<i>props</i>	Optional. Specifies other request parameters/properties specified as a dictionary.

 **Note:**

- For job definition creation, deletion and update, the value of the *jobName* argument can be a fully qualified name (that includes the package name); however, the default package `/oracle/apps/ess/custom` is always prefixed to it.
- As indicated by the previous note, you can only create/delete/update the job definitions present in the `/oracle/apps/ess/custom` namespace; prepackaged job definitions cannot be modified.
- If you do not provide a fully qualified path name for the *jobType* argument, a default package (`/oracle/as/ess/core/`) is prefixed to the name.

3.2.7.3 Examples

The following variables are used in these examples:

Variable	Value
HOSTING_APP	EssNativeHostingApp
JOB_DEF_PREDEF	<code>/oracle/apps/ess/TestJob</code>
JOB_DEF_NAME	TestJob_wlst
JOB_TYPE_NAME	JavaJobType
JOB_DESC	My WLST Test Defn
PARAMS	'SYS_retries':1, 'myParam': 'xyz'

- Show all job definitions present in the namespace of `HOSTING_APP`:
`manageSchedulerJobDefn('SHOW',HOSTING_APP)`
- Show the details of the job definition (`JOB_DEF_PREDEF`) part of `HOSTING_APP`:
`manageSchedulerJobDefn('SHOW',HOSTING_APP,jobName=JOB_DEF_PREDEF)`
- Create the new job definition `/oracle/apps/ess/custom/TestJob_wlst` in the namespace of `HOSTING_APP` with the *jobType* value of `/oracle/as/ess/core/JavaJobType`:
`manageSchedulerJobDefn('CREATE',HOSTING_APP,jobName=JOB_DEF_NAME,jobType=JOB_TYPE_NAME)`
- Create a new job definition named `/oracle/apps/ess/custom/TestJob_wlst` in the `HOSTING_APP` namespace with the *jobType* `/oracle/as/ess/core/JavaJobType` and a description with the value of `JOB_DESC`:

```
manageSchedulerJobDefn('CREATE',HOSTING_APP,jobName=JOB_DEF_NAME,jobType=JOB_TYPE_NAME,desc=JOB_DESC)
```

- **Create a new job definition named `/oracle/apps/ess/custom/TestJob_wlst` in the `HOSTING_APP` namespace with the `jobType` `/oracle/as/ess/core/JavaJobType` and a description and parameters defined by `JOB_DESC` and `PARAMS`:**

```
manageSchedulerJobDefn('CREATE',HOSTING_APP,jobName=JOB_DEF_NAME,jobType=JOB_TYPE_NAME,desc=JOB_DESC,props=PARAMS)
```

- **Update the job definition `/oracle/apps/ess/custom/JOB_DEF_NAME` with the new description `JOB_DESC`:**

```
manageSchedulerJobDefn('UPDATE',HOSTING_APP,jobName=JOB_DEF_NAME,desc=JOB_DESC)
```

- **Update the job definition `/oracle/apps/ess/custom/JOB_DEF_NAME` with the properties `PARAMS`:**

```
manageSchedulerJobDefn('UPDATE',HOSTING_APP,jobName=JOB_DEF_NAME,props=PARAMS)
```

- **Delete the job definition `/oracle/apps/ess/custom/JOB_DEF_NAME` in `HOSTING_APP`:**

```
manageSchedulerJobDefn('DELETE',HOSTING_APP,jobName=JOB_DEF_NAME)
```

- **Customize the prepackaged job definition `JOB_DEF_PREDEF` in `HOSTING_APP` with the customizable properties `SYS_retries` and `SYS_priority`:**

```
manageSchedulerJobDefn('CUSTOMIZE',HOSTING_APP,jobName=JOB_DEF_PREDEF,props={'SYS_retries':2,'SYS_priority':3})
```

3.2.7.4 Related Shell Script

[essManageJobDefn](#)

3.2.8 Manage Oracle Enterprise Scheduler Schedule Definitions

Command Category : Application Management Commands

Use with WLST: Online

3.2.8.1 Description

Manage (show, create, delete, update) a schedule definition in Oracle Enterprise Scheduler.

If *operation* is set to `SHOW` and if *schName* is specified, the command shows the details of this particular schedule definition; otherwise, the command shows a list of all schedule definitions that are part of the application. The command returns 0 or -1 depending on whether the operation succeeds or fails.

If *operation* is set to `CREATE`, the command returns the `MetadataObjectID` of the newly created schedule definition.

If *operation* is set to `DELETE` or `UPDATE`, the command returns 0 or -1, depending on whether the operation succeeds or fails.

3.2.8.2 Syntax

```
manageSchedulerSchedule(operation,appName,[schName],[schDesc],
[begin_time],[frequency],[interval],[count],[end_time],[month],[week],
[day],[date])
```

Argument	Definition
<i>operation</i>	<p>Specifies which of the following operations to perform on the job:</p> <ul style="list-style-type: none"> • SHOW (S) Displays the list of schedule definitions that are part of a particular application. If this operation is used, the <i>appName</i> argument is required and <i>schName</i> is optional. • CREATE (C) Creates a new schedule definition in the namespace of the relevant application (supplied as a parameter) in MDS. If this operation is used, the <i>appName</i>, <i>schName</i>, and <i>frequency</i> arguments are required and <i>schDesc</i> is optional. • DELETE (D) Deletes the specified job definition from the MDS. If this operation is used, the <i>appName</i> and <i>schName</i> arguments are required. • UPDATE (U) Updates an existing schedule definition with the specified property values (existing parameters persist if not overridden). If this operation is used, the <i>appName</i>, <i>schName</i>, and one or both of the <i>schDesc</i> and <i>frequency</i> arguments are required.
<i>appName</i>	Specifies the logical name of the job's hosting application.
<i>schName</i>	Optional. Specifies the name of the schedule definition. For the create, delete, and update operations, a default package named <i>/oracle/apps/ess/custom/</i> is prepended to the name provided.
<i>schDesc</i>	Optional. Specifies a description for the schedule definition when creating or updating a definition.
<i>begin_time</i>	Optional. Specifies the start time for a recurring event. The format is: <i>HH:MM:SS:DD:MM:YYYY</i>
<i>frequency</i>	<p>Optional. Frequency of recurrence. Valid values are: SECOND, MINUTE, HOUR, DAY, WEEK, MONTH, YEAR</p> <p>If you use any of the following options: SECOND, MINUTE, HOUR, DAY, WEEK, you must use the following arguments: <i>interval</i>, <i>begin_time</i>, [<i>count</i> <i>end_time</i>].</p> <p>If you use the MONTH option, you must use the following arguments [(<i>week day</i>) <i>date</i>] [<i>begin_time</i>] [<i>count</i> <i>end_time</i>].</p> <p>If you use the YEAR option, you must use the following arguments: [<i>month</i> {(<i>week day</i>) <i>date</i>}] [<i>begin_time</i>] [<i>count</i> <i>end_time</i>].</p>
<i>interval</i>	Optional. Specifies the repeat interval used with the <i>frequency</i> option. This is an integer value with a default value of 1.

Argument	Definition
<i>count</i>	Optional. Specifies the maximum number of repetitions for a request. This value is an integer.
<i>end_time</i>	Optional. Specifies the end time for recurrence. Format: <i>HH:MM:SS:DD:MM:YYYY</i>
<i>month</i>	Optional. Specifies the month of the year if <i>frequency</i> is set to YEAR. Valid values are 1-12.
<i>week</i>	Optional. Specifies the week of a month. Valid values are 1-6 and LAST.
<i>day</i>	Optional. Specifies the day of a week. Valid values are 1-7, where 1 represents Monday.
<i>date</i>	Optional. Specifies a day of month. Valid values are 1-31 and LAST.

3.2.8.3 Examples

The following variables are used in these examples:

Variable	Value
HOSTING_APP	EssNativeHostingApp
PREDEF_SCHEDULE	/oracle/apps/ess/demo/seeded/MyPredefinedSch
SCHEDULE_NAME	MyWlstSchedule
SCH_DESC	Description for WLST test schedule

- Show all schedule definitions present in the namespace of `HOSTING_APP`:

```
manageSchedulerSchedule('SHOW',HOSTING_APP)
```
- Show the details of the `HOSTING_APP` schedule definition `PREDEF_SCHEDULE`:

```
manageSchedulerSchedule('SHOW',HOSTING_APP,schName=PREDEF_SCHEDULE)
```
- Create a new recurring schedule definition named `/oracle/apps/ess/custom/MyWlstSchedule` in the namespace of `HOSTING_APP` and with the description `SCH_DESC`. The recurring schedule is:
 - Occur every 2 minutes
 - Start at 21:10:30 on May 03, 2012
 - Complete 3 iterations

```
manageSchedulerSchedule('CREATE',HOSTING_APP,schName=SCHEDULE_NAME,
schDesc=SCH_DESC,frequency='MINUTE',interval=2,count=3,
begin_time='21:10:30:03:05:12')
```
- Create a new recurring schedule definition named `/oracle/apps/ess/custom/MyWlstSchedule` in the namespace of `HOSTING_APP`. The recurring schedule is:
 - Occur every 2 months on Thursday of the 5th week (if applicable)
 - Start at 21:10:30 on May 03, 2012
 - Continue until 21:10:30 on May 04, 2013

```
manageSchedulerSchedule('CREATE',HOSTING_APP,schName=SCHEDULE_NAME,
frequency='MONTH',interval=2, begin_time='21:10:30:03:05:12', week=5, day=4,
end_time='21:10:30:04:05:13')
```

- Create a new recurring schedule definition named `/oracle/apps/ess/custom/MyWlstSchedule` in namespace of `HOSTING_APP`. The recurring schedule is:
 - Occur every 3 years, on the last day of May (May 31st)
 - Start at 21:10:30 on May 03, 2012
 - Complete 5 iterations

```
manageSchedulerSchedule('CREATE',HOSTING_APP,schName=SCHEDULE_NAME,
frequency='YEAR',interval=3,begin_time='21:10:30:03:05:12',month=5,date='LAST',count=5)
```

- Create a new recurring schedule definition named `/oracle/apps/ess/custom/MyWlstSchedule` in the namespace of `HOSTING_APP`. The recurring schedule is: occur hourly, starting now.

```
manageSchedulerSchedule('CREATE',HOSTING_APP,schName=SCHEDULE_NAME,
frequency='HOURLY')
```

- Update the schedule definition named `/oracle/apps/ess/custom/SCHEDULE_NAME` with the description `SCH_DESC`. The recurring schedule remains unchanged.

```
manageSchedulerSchedule('UPDATE',HOSTING_APP,schName=SCHEDULE_NAME,schDesc=SCH_DESC)
```

- Update the schedule definition named `/oracle/apps/ess/custom/SCHEDULE_NAME` with the following new recurring schedule: Occur every minute.

```
manageSchedulerSchedule('UPDATE',HOSTING_APP,schName=SCHEDULE_NAME,
frequency='MINUTE')
```

- Delete the `HOSTING_APP` schedule definition named `/oracle/apps/ess/custom/SCHEDULE_NAME`.

```
manageSchedulerSchedule('DELETE',HOSTING_APP,schName=SCHEDULE_NAME)
```

3.2.8.4 Related Shell Script

[essManageSchedule](#)

3.3 Oracle Enterprise Scheduler Batch Delete Requests

Job request data is kept as records in the Oracle Enterprise Scheduler runtime store tables. These job requests records are not removed from the runtime store until they are physically purged by a database administrator using SQL purge scripts. A request must be logically deleted before it can be physically purged. Oracle Enterprise Scheduler provides two ways to delete a request, either by a user performing a delete operation for a specific request, or by an Oracle Enterprise Scheduler administrator submitting a batch delete request.

The batch delete feature provides a way for an Oracle Enterprise Scheduler administrator to delete requests in a batch fashion. An Oracle Enterprise Scheduler administrator defines a set of delete criteria and submits a batch delete request using the Oracle Enterprise Scheduler runtime MBean interface. The delete criteria for the batch job are specified in the form of application request parameters. The submitted batch delete request uses a pre-defined job definition for a Java executable that is

supplied by Oracle Enterprise Scheduler. The batch delete job definition metadata is defined in the ESSAPP metadata repository.

When the batch delete job runs, it determines which completed absolute parent and instance parent requests satisfy the delete criteria specified for that batch job request and deletes the request hierarchy for those requests. The batch delete job is executed in the context of ESSAPP.

3.3.1 The batchDeleteSchedulerRequest Command

Command Category: Application Management Commands

Use with WLST: Online

3.3.1.1 Description

Submits a request for a batch delete job. You can use either a pre-defined schedule (in MDS) or specify a one-time execution time. If neither of these is specified, the request starts immediately. The request parameters are used to specify the delete criteria.

When the batch delete job runs, it determines which completed absolute parent and instance parent requests satisfy the delete criteria specified for that batch job request. The batch delete job deletes the request hierarchy for those requests.

3.3.1.2 Syntax

```
batchDeleteSchedulerRequests([desc],[schId],[start],[end],[params])
```

Argument	Definition
<i>desc</i>	Optional. Specifies a description for the batch delete request.
<i>schId</i>	Optional. Specifies the fully qualified name of the schedule (pre-defined) to use. The default package, if none is specified, is: /oracle/as/ess/essapp/custom/.
<i>start</i>	Optional. Specifies the time to start the request. Format: <i>HH:MM:SS:DD:MM:YYYY</i>
<i>end</i>	Optional. The time the request should terminate. Format: <i>HH:MM:SS:DD:MM:YYYY</i>
<i>params</i>	Optional. Request parameters specified as a dictionary. Batch Delete Parameters describes the parameters.

3.3.1.2.1 Batch Delete Parameters

A batch delete job uses application request parameters to specify the batch delete criteria and basic result information for that delete job. The class `oracle.as.scheduler.job.BatchDeleteProperty.java` defines constants for the batch delete parameters. The delete criteria parameters are described in the following two tables.

Parameter	Data Type	Constants
CriteriaApplication	String	BatchDeleteProperty.CRITERIA_APPLICATION
CriteriaProduct	String	BatchDeleteProperty.CRITERIA_PRODUCT
CriteriaJobDefn	String	BatchDeleteProperty.CRITERIA_JOBDEFN
CriteriaJobExecType	String	BatchDeleteProperty.CRITERIA_JOBEXECTYPE
CriteriaSubmitUser	String	BatchDeleteProperty.CRITERIA_SUBMIT_USER
CriteriaMinimumAge	Integer	BatchDeleteProperty.CRITERIA_MINIMUM_AGE
CriteriaRetentionAgeSuccess	Integer	BatchDeleteProperty.CRITERIA_RETENTION_AGE_SUCCESS
CriteriaRetentionAgeError	Integer	BatchDeleteProperty.CRITERIA_RETENTION_AGE_ERROR
CriteriaProcessLimit	Integer	BatchDeleteProperty.CRITERIA_RETENTION_AGE_WARNING
CriteriaRetentionAgeCancel	Integer	BatchDeleteProperty.CRITERIA_RETENTION_AGE_CANCEL
CriteriaProcessLimit	Long	BatchDeleteProperty.CRITERIA_PROCESS_LIMIT

Parameter	Default Value	Description
CriteriaApplication	null (none)	Specifies the application for which requests are to be deleted. If null, this parameter is ignored and is not used as part of the delete criteria.
CriteriaProduct	null (none)	Specifies the product for which requests are to be deleted. If null, this parameter is ignored and is not used as part of the delete criteria.
CriteriaJobDefn	null (none)	Specifies the string representation of the job definition MetadataObjectId for which requests are to be deleted. If null, this parameter is ignored and is not used as part of the delete criteria.
CriteriaJobExecType	null (none)	Specifies the string representation of the job execution type for which requests are to be deleted. If null, this parameter is ignored and is not used as part of the delete criteria.
CriteriaSubmitUser	null (none)	Specifies the submitting user for which requests are to be deleted. If null, this parameter is ignored and is not used as part of the delete criteria.
CriteriaMinimumAge	null (none)	Specifies a minimum age, in days, before a request that completed is eligible for deletion.
CriteriaRetentionAgeSuccess	null (none)	Specifies a minimum age, in days, before a request that completed as success is eligible for deletion. If this parameter value is null, then the value of CriteriaMinimumAge is used.

Parameter	Default Value	Description
CriteriaRetentionAgeError	null (none)	Specifies a minimum age, in days, before a request that completed as a failure is eligible for deletion. If this parameter value is null, then the value of <code>CriteriaMinimumAge</code> is used.
CriteriaProcessLimit	null (none)	Specifies a minimum age, in days, before a request that completed with a warning is eligible for deletion. If this parameter value is null, then the value of <code>CriteriaMinimumAge</code> is used.
CriteriaRetentionAgeCancel	null (none)	Specifies a minimum age, in days, before a request that was canceled is eligible for deletion. If this parameter value is null, then the value of <code>CriteriaMinimumAge</code> is used.
CriteriaProcessLimit	null (none)	Specifies the maximum number of requests to be processed by the batch delete job. If null or zero (0), there is no limit. The default value is zero (0).

3.3.1.3 Examples

- Delete all purgeable requests:

```
batchDeleteSchedulerRequests()
```

- Delete all purgeable requests in the application

ESS_NATIVE_HOSTING_APP_LOGICAL_NAME:

```
batchDeleteSchedulerRequests(desc='My purge for ESS
NativeApp',params={'CriteriaApplication':'ESS_NATIVE_HOSTING_APP_LOGICAL_NAME'})
```

- Delete all purgeable requests in the application

ESS_NATIVE_HOSTING_APP_LOGICAL_NAME for which the job definition is /oracle/apps/ess/custom/MyDef:

```
batchDeleteSchedulerRequests(desc='My purge for ESS
NativeApp',params={'CriteriaApplication':'ESS_NATIVE_HOSTING_APP_LOGICAL_NAME','C
riteriaJobDefn':'JobDefinition://oracle/apps/ess/custom/MyDef'})
```

- Delete all purgeable requests in the application

ESS_NATIVE_HOSTING_APP_LOGICAL_NAME for which the job type is JAVA_TYPE:

```
batchDeleteSchedulerRequests(desc='My purge for ESS NativeApp',
params={'CriteriaApplication':'ESS_NATIVE_HOSTING_APP_LOGICAL_NAME',
'CriteriaJobExecType':'JAVA_TYPE'})
```

- Submit batch delete job request using the schedule /oracle/as/ess/essapp/custom/WeeklySch:

```
batchDeleteSchedulerRequests(desc='Purge using WeekSch',schId='/oracle/as/ess/
essapp/custom/WeeklySch')
```

3.3.1.4 Related Shell Script

[essBatchDeleteRequests](#)

3.4 Work Allocation Commands

Table 3-2 shows the WLST commands Oracle Enterprise Scheduler provides for work allocation.

Table 3-2 Oracle Enterprise Scheduler Work Allocation Commands

Command	Description
addProcessorBinding	Bind a work assignment
createWorkshift	Create a custom workshift
deleteWorkAssignment	Delete work assignment metadata
deleteWorkshift	Delete workshift metadata
queryProcessorBindings	Query all processor bindings
queryWorkAssignments	Query work assignment metadata
queryWorkSchedules	Query schedule metadata for work allocation
queryWorkshifts	Query workshift metadata
removeProcessorBinding	Remove a processor binding
updateWorkshift	Update workshift resources

3.4.1 addProcessorBinding

Command Category: Application Management Commands

Use with WLST: Online

3.4.1.1 Description

Binds a work assignment to the current Oracle Enterprise Scheduler processor.

3.4.1.2 Syntax

```
addProcessorBinding(workAssignmentName,server=serverName,
[isExclusive=None][scope])
```

Argument	Definition
<i>workAssignmentName</i>	Specifies the name of the work assignment to bind.
<i>server</i>	Specifies the name of the server to which to bind.
<i>isExclusive</i>	Optional. Specifies whether the binding is exclusive (<code>true</code>). The default value is <code>false</code> .

Argument	Definition
<i>scope</i>	Optional. Can be <code>server</code> , <code>group</code> , or <code>global</code> . The default value is <code>server</code> . <ul style="list-style-type: none"> <code>server</code> binds to the specified server. <code>group</code> binds to the processing group of the specified server. <code>global</code> binds globally for all current and future servers in the isolation group.

3.4.1.3 Examples

- Bind a work assignment to a processor:

```
addProcessorBinding('mypkg/MyTestWA', server='ess_server1')
```
- Bind a work assignment to a processor in exclusive mode.

```
addProcessorBinding('SimpleApp', server='ess_server1', isExclusive='true')
```
- Bind a work assignment to a processor globally.

```
addProcessorBinding('SimpleApp', server='ess_server1', scope='global')
```

3.4.2 createWorkshift

Command Category: Application Management Commands

Use with WLST: Online

3.4.2.1 Description

Creates a custom workshift that specifies a time window and the resources to be used during that window.

3.4.2.2 Syntax

```
createWorkshift(workshiftName, alloc=allocation,
plsqlLimit=limit, asyncJavaLimit=limit, [description=None],
[schedName=None], [duration=None])
```

Argument	Definition
<i>workshiftName</i>	Specifies the name of the workshift to create.
<i>alloc</i>	Specifies the maximum number of threads that can be allocated on each server instance.
<i>plsqlLimit</i>	Specifies the maximum number of PL/SQL jobs that can be active in the isolation group. Use <code>-1</code> to specify no limit.
<i>asyncJavaLimit</i>	Specifies the number of async Java jobs that can be active in the isolation group. Use <code>-1</code> to specify no limit.
<i>description</i>	Optional. Specifies a description for the workshift.

Argument	Definition
<i>schedName</i>	Optional. Specifies the name of the schedule to use for the workshift. <ul style="list-style-type: none"> If the schedule name is specified, the <i>duration</i> argument is required. If the schedule name is <i>not</i> specified, the workshift is 24x7
<i>duration</i>	Specifies the number of minutes the workshift is active each time the schedule runs. Required if the <i>schedName</i> argument is used.

3.4.2.3 Example

- Create a 24x7 workshift in a specific custom package:

```
createWorkshift('mypkg/SampleWorkshift', alloc=5, plsqlLimit=15,
asyncJavaLimit=10)
```

3.4.3 deleteWorkAssignment

Command Category: Application Management Commands

Use with WLST: Online

3.4.3.1 Description

Deletes an Oracle Enterprise Scheduler work assignment in the custom namespace.

Note:

A work assignment that is bound cannot be deleted.

3.4.3.2 Syntax

```
deleteWorkAssignment(workAssignmentName)
```

Argument	Definition
<i>workAssignmentName</i>	Specifies the name of the custom work assignment to delete.

3.4.3.3 Example

- Delete a custom work assignment:

```
deleteWorkAssignment('MyTestWA')
```

3.4.4 deleteWorkshift

Command Category: Application Management Commands

Use with WLST: Online

3.4.4.1 Description

Deletes an Oracle Enterprise Scheduler workshift in the custom namespace.

**Note:**

A workshift that is in use by an on-board tenant or any bound work assignment cannot be deleted.

3.4.4.2 Syntax

```
deleteWorkshift(workshiftName)
```

Argument	Definition
<i>workshiftName</i>	Specifies the name of the custom workshift to delete.

3.4.4.3 Example

- Delete a custom workshift:

```
deleteWorkshift('mypkg/TenantWorkshift1')
```

3.4.5 queryProcessorBindings

Command Category: Application Management Commands

Use with WLST: Online

3.4.5.1 Description

Gets processor bindings.

3.4.5.2 Syntax

```
queryProcessorBindings(server=serverName)
```

Argument	Definition
<i>server</i>	Specifies the name of the server to query.

3.4.5.3 Example

- Get all bindings:

```
queryProcessorBindings(server='ess_server1')
```

3.4.6 queryWorkAssignments

Command Category: Application Management Commands

Use with WLST: Online

3.4.6.1 Description

Gets all custom and internal work assignments. Prints the work assignment id. Details include workshift IDs and specialization.

3.4.6.2 Syntax

```
queryWorkAssignments([name=None],[package=None],[options=None])
```

Argument	Definition
<i>name</i>	Optional. Specifies a work assignment name.
<i>package</i>	Optional. Specifies a work assignment package.
<i>options</i>	Optional. Comma-separated list of one or more of the following options: <ul style="list-style-type: none"> • <i>nameLike</i> Query for a name that contains the specified name. • <i>pkgLike</i> Query for a package that contains the specified package. • <i>details</i> Print additional details.

3.4.6.3 Examples

- Print all work assignments:

```
queryWorkAssignments()
```
- Print details for work assignments whose name contains "Simple":

```
queryWorkAssignments(name='Simple', options='nameLike,details')
```

3.4.7 queryWorkSchedules

Command Category: Application Management Commands

Use with WLST: Online

3.4.7.1 Description

Gets all schedules for work allocation and prints the schedule ID. Details include recurrence, exclusions, inclusions.

3.4.7.2 Syntax

```
queryWorkSchedules([name=None],[package=None],[options=None])
```

Argument	Definition
<i>name</i>	Optional. Specifies the schedule name.
<i>package</i>	Optional. Specifies the schedule package.

Argument	Definition
<i>options</i>	Optional. Comma-separated list of one or more of the following options: <ul style="list-style-type: none"> • <code>nameLike</code> Query for a name that contains the specified name. • <code>pkgLike</code> Query for a package that contains the specified package. • <code>details</code> Print additional details.

3.4.7.3 Examples

- Get all schedules:
`queryWorkSchedules()`
- Get details for an hourly schedule:
`queryWorkSchedules(name='Hourly', options='details')`

3.4.8 queryWorkshifts

Command Category: Application Management Commands

Use with WLST: Online

3.4.8.1 Description

Gets all workshifts and prints the workshift ID. Details include thread allocation and async limits.

3.4.8.2 Syntax

```
queryWorkshifts([name=None], [package=None], [options=None])
```

Argument	Definition
<i>name</i>	Optional. Specifies the workshift name.
<i>package</i>	Optional. Specifies the workshift package.
<i>options</i>	Optional. Comma-separated list of one or more of the following options: <ul style="list-style-type: none"> • <code>nameLike</code> Query for a name that contains the specified name. • <code>pkgLike</code> Query for a package that contains the specified package. • <code>details</code> Print additional details.

3.4.8.3 Examples

- Get all workshifts:
`queryWorkshifts()`

- Get workshifts whose package contains "test" and print the details:

```
queryWorkshifts(package='test', options='pkgLike,details')
```

3.4.9 removeProcessorBinding

Command Category: Application Management Commands

Use with WLST: Online

3.4.9.1 Description

Removes a local or global processor binding.

3.4.9.2 Syntax

```
removeProcessorBinding(workAssignmentName, server=serverName)
```

Argument	Definition
<i>workAssignmentName</i>	Specifies the name of the work assignment to unbind.
<i>server</i>	Specifies the name of the server from which to unbind.

3.4.9.3 Example

- Remove a binding:

```
removeProcessorBinding('mypkg/MyTestWA', server='ess_server1')
```

3.4.10 updateWorkshift

Command Category: Application Management Commands

Use with WLST: Online

3.4.10.1 Description

Update async limits and thread allocation for a custom workshift.

3.4.10.2 Syntax

```
updateWorkshift(workshiftName, [alloc=None], [plsqlLimit=None],  
[asyncJavaLimit=None])
```

Argument	Definition
<i>workshiftName</i>	Specifies the name of the workshift to update.
<i>alloc</i>	Optional. Specifies the maximum local thread allocation.
<i>plsqlLimit</i>	Optional. Specifies the maximum number of PL/SQL jobs that can be active in the isolation group. Use -1 to specify that there no limit.
<i>asyncJavaLimit</i>	Optional. Specifies the maximum number of async Java jobs that can be active in the isolation group. Use -1 to specify that there is no limit.

**Note:**

You must specify least one of the optional arguments.

3.4.10.3 Examples

- Update the PL/SQL limit for the workshift `mypkg/SampleWorkshift`.
`updateWorkshift('mypkg/SampleWorkshift', plsqlLimit=20)`
- Update both async limits for the workshift `mypkg/SampleWorkshift`.
`updateWorkshift('mypkg/SampleWorkshift', plsqlLimit=25, asyncJavaLimit=50)`

3.5 Oracle Enterprise Scheduler Diagnostic Dumps

Oracle Enterprise Scheduler provides a set of diagnostic dumps that facilitate the diagnosis of problems. These dumps are built on the Oracle Diagnostics Framework. Oracle Enterprise Scheduler problems are typically exposed when a request does not start or complete as expected. In such a scenario, the user can manually create Oracle Enterprise Scheduler diagnostic dumps and use the information in the dumps to help diagnose the problem. One way to easily create all Oracle Enterprise Scheduler diagnostic dumps is by creating an incident with a specific message ID, as described in [Creating an XXX Oracle Enterprise Scheduler Incident using WLSTXXX](#).

The following are the Oracle Enterprise Scheduler diagnostic dump-related commands. They are described in [The Dump Commands](#).

- `executeDump`
- `listDumps`
- `describeDump`

You control the behavior of the dump by specifying the dump name as an argument to the command. The valid dump names are described in [Table 3-3](#) and examples of their use are shown in [Dump Examples](#).

Table 3-3 Oracle Enterprise Scheduler Diagnostic Dumps

Dump Name	Description
<code>ess.applications</code>	Dumps all registered Oracle Enterprise Scheduler applications on every server in the isolation group. An application that is registered on multiple servers appears multiple times in this dump. For an application on a given server, you can check what version is registered, whether the application is active, and when it was last activated.
<code>ess.bindings</code>	Dumps current and inactive processor bindings for this instance, including the binding mode. An inactive binding that is completed has an expired schedule, and cannot become active again. For inactive bindings that are not completed, the dump shows the workshift that will next be active and when it will be active. This dump shows all completed bindings.

Table 3-3 (Cont.) Oracle Enterprise Scheduler Diagnostic Dumps

Dump Name	Description
<i>ess.processor</i>	Dumps current processor information for this instance. Dumps the processor state and active thread count, plus a list of enabled and deployed applications. The active thread count is the number of threads that are in use across all resources, including deactivated resources. For detailed information on work allocation, see dump <i>ess.workalloc</i> . If an application is not processing requests, make sure the processor is running and the application is both deployed and enabled on the server.
<i>ess.requests</i>	Dumps detailed information about all requests in the process group that are in a non-terminal state. Includes core and extended system properties and CP parameters.
<i>ess.servers</i>	Dumps server information, such as last check-in, for all active servers in the isolation group. Every server must check in at some interval (one minute by default). If a server hasn't checked in for over one minute, it may have a problem. After a server hasn't checked in for some period of time, Oracle Enterprise Scheduler may assume the server is down and fail over the requests that were being processed on that server.
<i>ess.workalloc</i>	Dumps detailed information on work assignments, including maximum resources and current allocations. The <code>whereClause</code> shows the effective specialization for the work assignment, which accounts for work assignments bound in exclusive mode. This dump may not show all completed bindings. Use <i>ess.bindings</i> to check if a binding is complete. If there are no bindings, this dump shows information on the default work assignment.
<i>ess.workunits</i>	Dumps all work units for this process group.

3.5.1 The Dump Commands

The following example shows how to start WLST and connect to the server:

```
$MW_HOME/oracle_common/common/bin/wlst.sh
...
Initializing WebLogic Scripting Tool (WLST) ...

Welcome to WebLogic Server Administration Scripting Shell

Type help() for help on available commands

wls:/offline> connect('weblogic','welcome1','localhost:7001')
Connecting to t3://localhost:7001 with userid weblogic ...
Successfully connected to Admin Server 'AdminServer' that belongs to domain
'base_domain'.

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.
```

```
wls:/base_domain/serverConfig>
```

**Note:**

You can connect to the admin server or a managed server. However, if you connect to the admin server, you must specify the managed server to use with each dump command, for example:

```
executeDump(name="ess.applications",
  appName="ESSAPP", server='ess_server1')
```

Once WLST is started and connected to the server, you can use the `executeDump`, `listDumps`, `describeDump` commands as shown in the following sections.

3.5.1.1 executeDump

The `executeDump` command creates a diagnostic dump based on the dump name and the application name.

3.5.1.1.1 Syntax

```
executeDump(name="dump_name", appName="app_name",
  [server="server_name"])
```

Argument	Definition
<i>name</i>	The dump name from Table 3-3 .
<i>appName</i>	The name of the application.
<i>server</i>	Optional. You can connect to the AdminServer or a managed server. If you connect to the AdminServer, you must use the <code>server</code> argument to specify the name of the managed server to use with each dump command.

3.5.1.2 listDumps

The `listDumps` command lists the available dumps for a given application.

3.5.1.2.1 Syntax

```
listDumps(appName="app_name")
```

Argument	Definition
<i>appName</i>	The name of the application.

3.5.1.3 describeDump

Lists help for a specific dump type.

3.5.1.3.1 Syntax

```
describeDump(name="dump_name")
```

Argument	Definition
<i>name</i>	The dump name from Table 3-3 .

3.5.2 Dump Examples

The following examples show how to use the diagnostic dumps listed in [Table 3-3](#).

- Dump all registered applications on every server in the isolation group. An application that is registered on more than one server appears multiple times in this dump. For an application on a given server, you can check what version is registered, whether the application is active, and when it was last activated.

```
executeDump(name="ess.applications",appName="ESSAPP")
```

- Dump processor bindings for this instance, including the binding mode. An inactive binding that is completed has an expired schedule, and it does not become active again. For inactive bindings that are not completed, the dump shows the workshift that will be active next and when it will be active. This dump shows all completed bindings.

```
executeDump(name="ess.bindings",appName="ESSAPP")
```

- Dump processor state and active thread count, plus list enabled and deployed applications. The active thread count is the number of threads that are in use across all resources, including deactivated resources. For more information about work allocation, see the entry for `ess.workalloc` in [Table 3-3](#).

If an application is not processing requests, make sure the processor is running and the application is both deployed and enabled on the server.

```
executeDump(name="ess.processor",appName="ESSAPP")
```

- Dump detailed information about all requests in the process group that are in a non-terminal state. Includes core and extended system properties and CP parameters.

```
executeDump(name="ess.requests",appName="ESSAPP")
```

- Dump all active servers in the isolation group. Every server must check in at an interval, one minute by default. If a server does not check in for over one minute, it may have a problem. If a server does not check in for some period of time, ESS may assume the server is down and fail over the requests that were being processed on that server.

```
executeDump(name="ess.servers",appName="ESSAPP")
```

- Dump detailed information about work assignments, including maximum resources and current allocations. The `whereClause` information shows the effective specialization for the work assignment, which accounts for work assignments bound in exclusive mode.

This dump may not show all completed bindings. To check whether a binding is complete, use the `ess.bindings` dump. If there are no bindings, this dump shows information about the default work assignment.

```
executeDump(name="ess.workalloc",appName="ESSAPP")
```

- Dump all work units in the process group.

```
executeDump(name="ess.workunits",appName="ESSAPP")
```

3.5.3 Creating an Oracle Enterprise Scheduler Incident

Oracle Enterprise Scheduler provides a convenient way to create an incident with all Oracle Enterprise Scheduler diagnostic dumps, using the message id `ESS-99999`. For example:

```
createIncident(messageId="ESS-99999",appName="ESSAPP", description="ESS  
incident with all dumps")
```

The incident files are located in the `ADRHome` directory, and you can use the `listADRHome()` command to list them.

Note that Oracle Enterprise Scheduler uses the following paths:

```
ADRBase = ${MW_HOME}/user_projects/domains/base_domain/servers/  
SERVER/adr
```

```
ADRHome = ${ADRBase}/diag/ofm/base_domain/SERVER
```

3.6 Oracle Enterprise Scheduler Convenience Scripts

To simplify execution, wrapper shell scripts (`.sh` on Linux/Unix and `.cmd` on Windows) are provided to invoke the native WLST commands. The scripts set environment properties such as `CLASSPATH` before using `wlst.sh` to invoke the WLST native commands. All the scripts are available in the `$ORACLE_HOME/bin` directory.

The `.sh` and `.cmd` commands connect to a WLS server before executing the corresponding WLST command. By default, the server connection details are read from a file named `server.properties` in the `HOME` directory of the user running the command (typically `/home/UserId` on Linux/Unix and `C:\Documents and Settings\UserId` on Windows). You can override the default file by specifying an alternate file or by providing explicit values (host, port, user name, password) on the command line. The host, port, user name and password values specified on the command line take precedence over the corresponding values in the files.

The connection details must be specified in the following syntax in the `server.properties` file or in any alternate file provided on the command line:

```
ADMIN_SERVER_HOST=host.example.com  
ADMIN_SERVER_PORT=7001  
ADMIN_USER_NAME=weblogic  
ADMIN_PASSWORD=weblogic1  
ESS_SERVER_NAME=ess_server1
```

In order to comply with manageability checklist requirements, the password cannot be supplied as a command-line argument while invoking shell scripts. It can either be part of the `server.properties` file or entered interactively. You can use one of the following methods to automate interactive password submission:

- Write the WLS administrator password in a file, and then use it to redirect input to command. For example:

```
sh essManageRuntimeConfig.sh .... < input.txt
```

- After the command finishes use "<<EOF". EOF can be any unique string other than the actual password. In the next line write the password, followed by EOF in the next line (or whatever terminating string you used after <<). For Example:

```
sh essManageRuntimeConfig.sh ... <<EOF
wls_admin_password
EOF
```

3.6.1 essManageRuntimeConfig

Manages Oracle Enterprise Scheduler runtime configuration parameters. The server connection parameters are specified using a file as described in [Oracle Enterprise Scheduler Convenience Scripts](#). However, the default values can be overridden by explicitly specifying them on the command line or specifying an alternate file using the `-f` option. The admin server password for WLS has to be provided either in a file or entered interactively.

3.6.1.1 Syntax

```
essManageRuntimeConfig.sh [-a | -m | -d] -A appname -t parameter_type [-n
name] [-v value] [-f filename] [-H hostname] [-P port] [-u user] [-s
server_name] [-h]
```

3.6.1.2 Options

The following table lists and describes the command options.

Option	Description
<code>-a, --add</code>	Adds a configuration parameter
<code>-m, --mod</code>	Modifies a configuration parameter
<code>-d, --del</code>	Deletes a configuration parameter.
<code>-A, --app <i>appname</i></code>	Specifies the name of the hosting application.
<code>-t, --type <i>parameter_type</i></code>	Specifies the type of the configuration parameter. Possible types are: <ul style="list-style-type: none"> • ESS - for Oracle Enterprise Scheduler parameters • APP - for user defined parameters
<code>-n, --name <i>name</i></code>	Specifies the name of the configuration parameter.
<code>-v, --val <i>value</i></code>	Specifies the value of the configuration parameter.
<code>-f, --file <i>filename</i></code>	Specifies the name of the file that contains the server connection parameters. If not specified, the default file <code>\$HOME/server.properties</code> is used.
<code>-H, --host <i>hostname</i></code>	Specifies the hostname or IP address of the WLS server.
<code>-P, --port <i>port</i></code>	Specifies the port number to connect.
<code>-u, --user <i>username</i></code>	Specifies the user name.
<code>-s, --serv <i>server_name</i></code>	Specifies the name of the server on which ESSAPP is running.
<code>-h</code>	Displays the command usage.

3.6.1.3 Associated Files

The following file is used with this command:

`server.properties`: Contains server connection information

3.6.1.4 Exit Values

The command exits with the following possible values:

- 0: Success
- -1: Error

3.6.1.5 Examples

- Add a user-defined parameter `foo` with value `bar` for the application `myapp`:

```
essManageRuntimeConfig.sh -a -A myapp -t APP -n foo -v bar
```
- Get the value of user-defined parameter `foo` for the application `myapp`:

```
essManageRuntimeConfig.sh -A myapp -t APP -n foo
```
- Get the list of all user-defined parameters for the application `myapp`:

```
essManageRuntimeConfig.sh -A myapp -t APP
```
- Modify the value of the user-defined parameter `foo` to `barone` for the application `myapp`:

```
essManageRuntimeConfig.sh -m -A myapp -t APP -n foo -v barone
```
- Delete the user-defined parameter `foo` for application `myapp`:

```
essManageRuntimeConfig.sh -d -t APP -A myapp -n foo
```
- Show all ESS parameters for `myapp`:

```
essManageRuntimeConfig.sh -t ESS -A myapp
```

3.6.1.6 Related Native Command

[Manage \(Add/Delete/Modify/Get\) Configuration Parameters](#)

3.6.2 essGetOutputContent

Retrieves the request log and output data files from the content store for all the specified request IDs. The server connection parameters are specified using a file as described in [Oracle Enterprise Scheduler Convenience Scripts](#). However, the default values can be overridden by explicitly specifying them on the command line or specifying an alternate file using the `-f` option. The admin server password for WLS has to be provided in a file or entered interactively.

3.6.2.1 Syntax

```
essGetOutputContent.sh [-t content_type] [-d dir] [-n lines] [-x disp] [-f  
filename] [-H hostname] [-P port] [-u user] [-s server_name] [-h]  
requestId1 ...
```

3.6.2.2 Options

The following table lists and describes the command options.

Option	Description
-t, --content <i>type</i>	Specifies one of the following types of content to get: <ul style="list-style-type: none"> LOG OUTPUT TEXT_OUTPUT BINARY_OUTPUT If this option is not specified, both log and output content are retrieved.
-d, --outDir <i>dir</i>	Specifies the directory where content is stored. The default is the current directory.
-n, --logLines <i>lines</i>	Specifies the number of lines of log to retrieve. The default is 1000.
-x, --logDisplay <i>disp</i>	Specifies where log lines are presented. The valid values are CONSOLE or FILE. The default is FILE.
-f, --file <i>filename</i>	Specifies the name of the file that contains the server connection parameters. If not specified, the default file \$HOME/server.properties is used.
-H, --host <i>hostname</i>	Specifies the hostname or IP address of the WLS server.
-P, --port <i>port</i>	Specifies the port number to connect.
-u, --user <i>username</i>	Specifies the user name.
-s, --serv <i>server_name</i>	Specifies the name of the server on which ESSAPP is running.
-h	Displays the command usage.

3.6.2.3 Associated Files

The following file is used with this command:

`server.properties`: Contains server connection information

3.6.2.4 Exit Values

The command exits with the following possible values:

- 0: Success
- 1: Error

3.6.2.5 Examples

- Get the request log for request ID 123:
`essGetOutputContent.sh -t LOG 123`
- Get all of the output from request 123:
`essGetOutputContent.sh -t OUTPUT 123`

- Get all the output of request 123 and save it in directory /tmp:
`essGetOutputContent.sh -t OUTPUT -d "/tmp" 123`
- Get the first 100 lines of the request log for request ID 123:
`essGetOutputContent.sh -t LOG -n 100 123`
- Get the request log and output data for requests 123 and 124 and save it in /tmp:
`essGetOutputContent.sh -d "/tmp" 123 124`

3.6.2.6 Related Native Command

[Get Log and Output Content of a Request](#)

3.6.3 essQueryRequests

Search and list requests based upon hosting application name, state or elapsed time of execution. This command can be used to find long running requests. The server connection parameters are specified using a file as described in [Oracle Enterprise Scheduler Convenience Scripts](#). However, the default values can be overridden by explicitly specifying them on the command line or specifying an alternate file using the `-f` option. The admin server password for WLS has to be provided in a file or entered interactively.

3.6.3.1 Syntax

```
essQueryRequests.sh [-a app_name] [-S state] [-d days] [-n hours] [-m
minutes] [-f filename] [-H hostname] [-P port] [-u user] [-s server_name]
[-h]
```

3.6.3.2 Options

The following table lists and describes the command options.

Option	Description
<code>-a, --appName app_name</code>	Specifies the hosting application name.

Option	Description
-S, --state <i>state</i>	Specifies one of the following request states: <ul style="list-style-type: none"> • WAIT • READY • RUNNING • COMPLETED • BLOCKED • HOLD • CANCELLING • EXPIRED • CANCELLED • ERROR • WARNING • SUCCEDED • PAUSED • PENDING_VALIDATION • VALIDATION_FAILED • SCHEDULE_ENDED • FINISHED • ERROR_AUTO_RETRY • ERROR_MANUAL_RECOVERY
-d, --days <i>days</i>	Specifies the running time in days.
-n, --hours <i>hours</i>	Specifies the running time in hours.
-m, --minutes <i>minutes</i>	Specifies the running time in minutes.
-f, --file <i>filename</i>	Specifies the name of the file that contains the server connection parameters. If not specified, the default file <code>\$HOME/server.properties</code> is used.
-H, --host <i>hostname</i>	Specifies the hostname or IP address of the WLS server.
-P, --port <i>port</i>	Specifies the port number to connect.
-u, --user <i>username</i>	Specifies the user name.
-s, --serv <i>server_name</i>	Specifies the name of the server on which ESSAPP is running.
-h	Displays the command usage.

3.6.3.3 Associated Files

The following file is used with this command:

`server.properties`: Contains server connection information

3.6.3.4 Exit Values

The command exits with the following possible values:

- 0: Success
- -1: Error

3.6.3.5 Examples

- Get all running requests:
`essQueryRequests.sh -S RUNNING`
- Get all cancelled requests:
`essQueryRequests.sh -S CANCELLED`
- Get all requests running for more than two days:
`essQueryRequests.sh -d 2`
- Get all requests running for more than 10 hours:
`essQueryRequests.sh -n 10`
- Get all requests running for the application `myapp`:
`essQueryRequests.sh -a myapp`
- Get all requests in the CANCELLING state for the application `myapp`:
`essQueryRequests.sh -a myapp -S CANCELLING`
- Get all requests that have been running for more than 2.5 days for the application `myapp`:
`essQueryRequests.sh -a myapp -d 2 -n 12`

3.6.3.6 Related Native Command

[Search and List Requests](#)

3.6.4 essManageRequests

Cancel, recover, or complete one or more requests given their request IDs. The server connection parameters are specified using a file as described in [Oracle Enterprise Scheduler Convenience Scripts](#). However, the default values can be overridden by explicitly specifying them on the command line or specifying an alternate file using the `-f` option. The admin server password for WLS has to be provided in a file or entered interactively.

3.6.4.1 Syntax

```
essManageRequests.sh [-c] [-r] [-C] [-g] [-S status] [-M message] [-f
filename] [-H hostname] [-P port] [-u user] [-s server_name] [-h]
requestId1 ...
```

3.6.4.2 Options

The following table lists and describes the command options.

Option	Description
<code>-c, --cancel</code>	Cancels request.
<code>-r, --recover</code>	Recovers a request

Option	Description
-C, --complete	Marks a request as completed.
-g, --getdetail	Gets a request's details.
-S, --asyncStatus <i>status</i>	When marking request as complete, specifies one of the following final disposition statuses: <ul style="list-style-type: none"> • SUCCESS • PAUSE • WARNING • ERROR • CANCEL • UPDATE • BIZ_ERROR • ERROR_MANUAL_RECOVERY
-M, --statusMessage <i>message</i>	Specifies a qualifying message that describes the operation when marking request as complete.
-f, --file <i>filename</i>	Specifies the name of the file that contains the server connection parameters. If not specified, the default file <code>\$HOME/server.properties</code> is used.
-H, --host <i>hostname</i>	Specifies the hostname or IP address of the WLS server.
-P, --port <i>port</i>	Specifies the port number to connect.
-u, --user <i>username</i>	Specifies the user name.
-s, --serv <i>server_name</i>	Specifies the name of the server on which ESSAPP is running.
-h	Displays the command usage.

3.6.4.3 Associated Files

The following file is used with this command:

`server.properties`: Contains server connection information

3.6.4.4 Exit Values

The command exits with the following possible values:

- 0: Success
- -1: Error

3.6.4.5 Examples

- Cancel requests 123 and 124:
`essManageRequests.sh -c 123 124`
- Recover request 123:
`essManageRequests.sh -r 123`
- Complete request 123:

```
essManageRequests.sh -C -S ERROR -M "Completed by Admin" 123
```

3.6.4.6 Related Native Command

Manage Requests

3.6.5 essManageServer

Start, stop or get the status of the Oracle Enterprise Scheduler application running on the server. Starting the Oracle Enterprise Scheduler application means to start the Oracle Enterprise Scheduler processor thread so that request processing can start. Stopping Oracle Enterprise Scheduler means to stop or quiesce the Oracle Enterprise Scheduler processor so that no new requests are processed. The server connection parameters are specified using a file as described in [Oracle Enterprise Scheduler Convenience Scripts](#). However, the default values can be overridden by explicitly specifying them on the command line or specifying an alternate file using the `-f` option. The admin server password for WLS has to be provided in a file or entered interactively.

3.6.5.1 Syntax

```
essManageServer.sh [-start] [stop] [-status] [-f filename] [-H hostname]
[-P port] [-u user] [-s server_name] [-h]
```

3.6.5.2 Options

The following table lists and describes the command options.

Option	Description
<code>-n, --start</code>	Starts the Oracle Enterprise Scheduler processor thread.
<code>-x, --stop</code>	Stops the Oracle Enterprise Scheduler processor thread.
<code>-S, --status</code>	Displays the current status of the Oracle Enterprise Scheduler processor thread.
<code>-f, --file <i>filename</i></code>	Specifies the name of the file that contains the server connection parameters. If not specified, the default file <code>\$HOME/server.properties</code> is used.
<code>-H, --host <i>hostname</i></code>	Specifies the hostname or IP address of the WLS server.
<code>-P, --port <i>port</i></code>	Specifies the port number to connect.
<code>-u, --user <i>username</i></code>	Specifies the user name.
<code>-s, --serv <i>server_name</i></code>	Specifies the name of the server on which ESSAPP is running.
<code>-h</code>	Displays the command usage.

3.6.5.3 Associated Files

The following file is used with this command:

`server.properties`: Contains server connection information

3.6.5.4 Exit Values

The command exits with the following possible values:

- 0: Success
- -1: Error

3.6.5.5 Examples

- Stop Oracle Enterprise Scheduler processor:

```
essManageServer.sh --stop
```

- Get the current state of the Oracle Enterprise Scheduler processor:

```
essManageServer.sh --status
```

3.6.5.6 Related Native Command

[Manage Oracle Enterprise Scheduler Servers](#)

3.6.6 essSubmitRequest

Submit a job request to ESSAPP for execution. It is assumed that the job definition exists in the MDS. The user can specify the job execution schedule in the following four ways:

- Immediate: The job is scheduled for immediate one-time execution.
- Once: The job is scheduled to run once at a fixed time specified by the user.
- Schedule: The name of a predefined schedule (in MDS) is provided and the job runs on that schedule.
- Recurring: The schedule can be created in an ad-hoc manner by providing relevant arguments through this command. This schedule is used for the request submission and it is stored in the MDS for future use. The storage location is based on the schedule name provided.

If no scheduling method is specified, immediate execution is used.

Note:

The job definition must be predefined, while the schedule may be defined in an ad-hoc manner using this command.

The `jobName` and `schName` options must be fully qualified names that includes the package name. If they are not fully qualified, the following package is used: `/oracle/apps/ess/custom/`.

The server connection parameters are specified using a file as described in [Oracle Enterprise Scheduler Convenience Scripts](#). However, the default values can be overridden by explicitly specifying them on the command line or specifying an alternate

file using the `-f` option. The admin server password for WLS has to be provided in a file or entered interactively.

3.6.6.1 Syntax

```
essSubmitRequest.sh appName jobName [-n reqNote] [-I | -O time | -R params
| -S schName] [-f filename] [-H hostname] [-P port] [-u user] [-s
server_name] [-h] [prop=value ...]
```

3.6.6.2 Options

The following table lists and describes the command options.

Option	Description
appName	Names the job's hosting application.
jobName	Names the job definition to be used for the current request.
-n, --note reqNote	Specifies a short description for the job request
-S, --schName schName	Names the predefined schedule definition.
-A, --RschName name	Names the schedule if you define the schedule using <code>-R</code> option.
-C, --schDesc desc	Specifies a description of the ad-hoc schedule being defined. The default value is "Custom Ad-Hoc Schedule".
-I, --immediate	Specifies that the job should be executed immediately. This is used as the default if no other schedule is specified.
-O, --o_time time	Specifies the time for one time execution. Format: HH:MM:SS:DD:MM:YYYY.
-R, --recurring	Specifies that the schedule recurs.
-b, --begin_time time	Specifies the start time for a recurring schedule. Format: HH:MM:SS:DD:MM:YYYY .
-F, --frequency value	Specifies the frequency of recurrence. Valid values: [SECOND,MINUTE,HOURL,DAY,WEEK,MONTH,YEAR]
-N, --freqNum num	Specifies an integer value that represents the frequency repeat interval.
-c, --count num	Specifies an integer value that is the maximum number of repetitions.
-e, --end_time time	Specifies the recurrence end time. Format: HH:MM:SS:DD:MM:YYYY.
-m, --month month	Defines the month of the year if the frequency is YEAR. Valid values are: [1-12].
-w, --week week	Specifies the week of a month. Valid values are: [1-6, LAST].
-d, --day day	Specifies the day of a week. Valid values are: [1-7], where 1 represents Monday.
-D, --date date	Specifies a specific day of month. Valid values are: [1-31, LAST].

Option	Description
<i>prop=value</i>	Specifies other request parameters/properties specified as: <i>prop1=val1 prop2=val2</i> and so on. These values should be added as the last options in the command.
<i>-f, --file filename</i>	Specifies the name of the file that contains the server connection parameters. If not specified, the default file <code>\$HOME/server.properties</code> is used.
<i>-H, --host hostname</i>	Specifies the hostname or IP address of the WLS server.
<i>-P, --port port</i>	Specifies the port number to connect.
<i>-u, --user username</i>	Specifies the user name.
<i>-s, --serv server_name</i>	Specifies the name of the server on which ESSAPP is running.
<i>-h</i>	Displays the command usage.

3.6.6.3 Associated Files

The following file is used with this command:

`server.properties`: Contains server connection information

3.6.6.4 Exit Values

The command exits with the following possible values:

- 0: Success
- -1: Error

3.6.6.5 Examples

The following values apply to all of the examples:

`HOSTING_APP = EssDemoApp`

`SCHEDULE_NAME= /oracle/apps/ess/custom/MySch`

`JOB_DEF_NAME= /oracle/apps/ess/demopackage/BasicJavaJob`

and server connection properties are defined in the `~/server.properties` file.

Note:

The `[prop-value ...]` values should be added as the last options in the command.

- Submit a request for immediate job execution (one-time only).:


```
essSubmitRequest.sh HOSTING_APP JOB_DEF_NAME
```
- Submit a request for one-time job execution at a specified time.:


```
essSubmitRequest.sh HOSTING_APP JOB_DEF_NAME -O 21:10:30:03:05:2012
```

- Submit a job request with a recurring schedule. The recurrence is defined as follows:
 - Execution recurs every 2 minutes
 - Starts at 21:10:30 on 03 May 2012
 - Completes three iterations
 - Additional request parameters are: `reqName=test` and `PRIORITY=10`. This recurrence will be persisted in the MDS repository for future use.

```
essSubmitRequest.sh HOSTING_APP JOB_DEF_NAME -R -A SCHEDULE_NAME -F MINUTE -N 2 -c 3 -b 21:10:30:03:05:2012 reqName=test PRIORITY=10
```

- Submit a job request using a predefined schedule:

```
essSubmitRequest.sh HOSTING_APP JOB_DEF_NAME -S SCHEDULE_NAME
```

- Submit a job request with a monthly recurring schedule. The description for the request is "Request with monthly schedule". The recurring schedule name and description are provided by `-A` and `-C` options. The recurrence is: Execution recurs every 2 months, on Thursday of the 5th week (if applicable), starting at 21:10:30 on 03 May 2012, until 21:10:30 on 04 May 2013:

```
essSubmitRequest.sh HOSTING_APP JOB_DEF_NAME -n "Request with monthly schedule" -R -A SCHEDULE_NAME -C Monthly_schedule -F MONTH -N 2 -b 21:10:30:03:05:2012 -w 5 -d 4 -e 21:10:30:04:05:2013
```

- Submit a job request with a yearly recurring schedule. The description for the request is "Request with yearly schedule". The recurring schedule name and description are provided by the `-A` and `-C` options. The recurrence is: Execution will recur every 3 years, on the last day of May, starting at 21:10:30 on 03 May 2012, for 5 iterations:

```
essSubmitRequest.sh HOSTING_APP JOB_DEF_NAME -n "Request with yearly schedule" -R -A SCHEDULE_NAME -C "Yearly_schedule" -F YEAR -N 3 -b 21:10:30:03:05:2012 -m 5 -D LAST -c 5
```

- Submit a job request with an hourly recurring schedule. Recurrence is: Execution repeats every hour starting at the current time.:

```
essQueryRequests.sh -a myapp -d 2 -n 12
```

3.6.6.6 Related Native Command

[Submit Job Requests to Oracle Enterprise Scheduler](#)

3.6.7 essManageJobDefn

Manage (show, create, delete, customize, update) an Oracle Enterprise Scheduler job definition.

The `show` option (`-S`) displays the list of job definitions that are part of a particular application. If a job name is specified it only shows the details of this job definition. The `create` option (`-N`) creates a new job definition in the namespace of the relevant application (supplied as a parameter) in MDS. The `delete` option (`-D`) deletes a job definition from the MDS. The `customize` option (`-C`) modifies the customizable properties of an existing job definition. The `update` option (`-U`) updates an existing job

definition with the desired property values (existing parameters persist if not overridden).

 **Note:**

- For job definition creation, deletion and update, the default package path `/oracle/apps/ess/custom/` is always prepended to the specified job name.
- You can only create, delete, or update the job definitions present in the `/oracle/apps/ess/custom` namespace. That means that prepackaged job definitions cannot be modified.
- You cannot customize job definitions present in the customer namespace `/oracle/apps/ess/custom`. Use the update option to modify those definitions.
- For the `jobType` specification, if you do not specify a fully qualified name, the default package `/oracle/as/ess/core/` is prepended to the name.

The server connection parameters are specified using a file as described in [Oracle Enterprise Scheduler Convenience Scripts](#). However, the default values can be overridden by explicitly specifying them on the command line or specifying an alternate file using the `-f` option. The admin server password for WLS has to be provided in a file or entered interactively.

3.6.7.1 Syntax

```
essManageJobDefn.sh appName [-j jobName] [-t jobType] [-d desc] -S | -N |
-D | -C | -U [-f filename] [-H hostname] [-P port] [-u user] [-s
server_name] [-h] [prop=value ...]
```

3.6.7.2 Options

The following table lists and describes the command options.

Option	Description
<i>appName</i>	Specifies the logical name of the hosting application for a job.
<code>-j, --jobName <i>jobDefn</i></code>	Specifies the name of the job definition. For the create, delete, and update operations, the default package <code>/oracle/apps/ess/custom/</code> is prepended to the specified name.
<code>-t, --jobType <i>jobType</i></code>	Specifies the name of the predefined job type to be used for creating the job definition. If the name does not contain package path, the following path is prepended to it: <code>/oracle/as/ess/core/</code> .
<code>-d, --desc <i>description</i></code>	Specifies a description of the job definition

Option	Description
-S, --show	Displays the list of job definitions that are part of a particular application. If a job name is specified it shows the details of only that job definition. Used with <i>appName</i> and -j.
-N, --create	Creates a new job definition. Used with <i>appName</i> , -j, -t, -d and <i>prop=value</i> .
-D, --delete	Deletes an existing job definition. Used with <i>appName</i> and -j.
-C, --customize	Modifies customizable properties of the job definition. Properties can be: <ul style="list-style-type: none"> • SYS_retries • SYS_priority • SYS_requestCategory • SYS_request_timeout • enableTrace • enableTimeStatistics Used with <i>appName</i> , -j option, <i>prop=value</i> .
-U, --update	Updates an existing job definition. This option replaces the current definition with a new one in which existing parameters persist if not overridden. Used with <i>appName</i> , -j and either -d or <i>prop=value</i> .
<i>prop=value</i>	Specifies other request parameters/properties specified as: <i>prop1=val1 prop2=val2</i> and so on.
-f, --file <i>filename</i>	Specifies the name of the file that contains the server connection parameters. If not specified, the default file <code>\$HOME/server.properties</code> is used.
-H, --host <i>hostname</i>	Specifies the hostname or IP address of the WLS server.
-P, --port <i>port</i>	Specifies the port number to connect.
-u, --user <i>username</i>	Specifies the user name.
-s, --serv <i>server_name</i>	Specifies the name of the server on which ESSAPP is running.
-h	Displays the command usage.

3.6.7.3 Associated Files

The following file is used with this command:

`server.properties`: Contains server connection information

3.6.7.4 Exit Values

The command exits with the following possible values:

- 0: Success
- -1: Error

3.6.7.5 Examples

The following values apply to all of the examples:

```
HOSTING_APP = ESS_NATIVE_HOSTING_APP_LOGICAL_NAME
```

```
JOB_DEF_PREDEF= /oracle/apps/ess/TestJob
```

```
JOB_DEF_NAME = TestJob_wlst
```

```
JOB_TYPE_NAME = JavaJobType
```

```
JOB_DESC = My WLST Test Defn
```

and server connection properties are defined in the `~/server.properties` file.

- Show all job definitions present in namespace of `HOSTING_APP`:

```
essManageJobDefn.sh HOSTING_APP -S
```
- Show details of the job definition `JOB_DEF_PREDEF` part of `HOSTING_APP`:

```
essManageJobDefn.sh HOSTING_APP -S -j JOB_DEF_PREDEF
```
- Create a new job definition named `/oracle/apps/ess/custom/TestJob_wlst` in the namespace of `HOSTING_APP` with the job type `/oracle/as/ess/core/JavaJobType`:

```
essManageJobDefn.sh HOSTING_APP -N -j JOB_DEF_NAME -t JOB_TYPE_NAME
```
- Create a new job definition named `/oracle/apps/ess/custom/TestJob_wlst` in namespace of `HOSTING_APP` with the job type `/oracle/as/ess/core/JavaJobType` and with the description `JOB_DESC`:

```
essManageJobDefn.sh HOSTING_APP -N -j JOB_DEF_NAME -t JOB_TYPE_NAME -d JOB_DESC
```
- Create a new job definition named `/oracle/apps/ess/custom/TestJob_wlst` in the namespace of `HOSTING_APP` with the job type `/oracle/as/ess/core/JavaJobType` and with the description `JOB_DESC` and the properties `SYS_retries` and `myParam`:

```
essManageJobDefn.sh HOSTING_APP -N -j JOB_DEF_NAME -t JOB_TYPE_NAME -d JOB_DESC  
SYS_retries=1 myParam=xyz
```
- Update the job definition `/oracle/apps/ess/custom+JOB_DEF_NAME` with the new description `JOB_DESC`:

```
essManageJobDefn.sh HOSTING_APP -U -j JOB_DEF_NAME -d JOB_DESC
```
- Update the job definition `/oracle/apps/ess/custom+JOB_DEF_NAME` with the properties `SYS_retries` and `myParam`:

```
essManageJobDefn.sh HOSTING_APP -U -j JOB_DEF_NAME SYS_retries=1 myParam=xyz
```
- Delete the job definition `/oracle/apps/ess/custom+JOB_DEF_NAME` in `HOSTING_APP`:

```
essManageJobDefn.sh HOSTING_APP -D -j JOB_DEF_NAME
```
- Customize the prepackaged job definition `JOB_DEF_PREDEF` in `HOSTING_APP` with the customizable properties `SYS_retries` & `SYS_priority`.:

```
essManageJobDefn.sh HOSTING_APP -C -j JOB_DEF_PREDEF SYS_retries=2 SYS_priority=3
```

3.6.7.6 Related Native Command

[Manage Oracle Enterprise Scheduler Job Definitions](#)

3.6.8 essManageSchedule

Manage (show, create, delete, customize and update) an Oracle Enterprise Scheduler schedule definition.

The `show` option (-S) displays the list of schedule definitions that are part of a particular application. If a schedule name is specified, it one shows the details for that schedule definition.

The `create` option (-C) creates a new schedule definition in the namespace of the relevant application (supplied as a parameter) in the MDS.

The `delete` option (-D) deletes the specified schedule definition from the MDS.

The `update` option (-U) updates an existing schedule definition with the specified property values (existing parameters persist if not overridden).

The server connection parameters are specified using a file as described in [Oracle Enterprise Scheduler Convenience Scripts](#). However, the default values can be overridden by explicitly specifying them on the command line or specifying an alternate file using the `-f` option. The admin server password for WLS has to be provided in a file or entered interactively.



Note:

The default package name: `/oracle/apps/ess/custom/` is always prepended to the schedule name specified with the `-n` option.

3.6.8.1 Syntax

```
essManageSchedule.sh appName [-n schName] [-d desc] -S | -C | -D | -U -F
value [-I interval] [-b begin_time] [-c count | -e end_time][ -f filename]
[-H hostname] [-P port] [-u user] [-s server_name]
```

3.6.8.2 Options

The following table lists and describes the command options.

Option	Description
<i>appName</i>	Specifies the logical name of the hosting application.
<code>-n, --schName schName</code>	Specifies the name of the schedule definition. For the create, delete, and update operations, the default package <code>/oracle/apps/ess/custom/</code> is prepended to the specified name.
<code>-d, --schDesc description</code>	Specifies a description of the schedule definition

Option	Description
-S, --show	Displays the list of schedule definitions that are part of a particular application. If a schedule name is specified it shows the details of only that schedule definition. Used with <i>appName</i> , -n and -d.
-C, --create	Creates a new schedule definition. Used with <i>appName</i> , -j, -t, -d and <i>prop=value</i> .
-D, --delete	Deletes an existing schedule definition. Used with <i>appName</i> and -n.
-U, --update	Updates an existing schedule definition. This option replaces the current definition with a new one in which existing parameters persist if not overridden. Used with <i>appName</i> , -n and either -d or -F.
-b, --begin_time <i>time</i>	Specifies the recurrence start time. Format: HH:MM:SS:DD:MM:YYYY
-F, --frequency <i>value</i>	Specifies the frequency of recurrence. Valid values: SECOND, MINUTE, HOUR, DAY, WEEK, MONTH, YEAR Possible options: -F <i>value</i> [-I <i>interval</i>] [-b <i>begin_time</i>] [-c <i>count</i> -e <i>end_time</i>] (if <i>value</i> is SECOND, MINUTE, HOUR, DAY, WEEK otherwise, one of the following) -F MONTH [{-w <i>week</i> -d <i>day</i> } {-D <i>date</i> }] [-b <i>begin_time</i>] [-c <i>count</i> -e <i>end_time</i>] -F YEAR [-m <i>month</i> ({-w <i>week</i> -d <i>day</i> } {-D <i>date</i> })] [-b <i>begin_time</i>] [-c <i>count</i> -e <i>end_time</i>]
-I, --interval <i>num</i>	Specifies an integer value that is used as the repeat interval for the -F option.
-c, --count <i>num</i>	Specifies an integer value that is the maximum number of repetitions.
-e, --end_time <i>time</i>	Specifies the recurrence end time. Format: HH:MM:SS:DD:MM:YYYY.
-m, --month <i>month</i>	Defines the month of the year if the frequency is YEAR. Valid values are: [1-12].
-f, --file <i>filename</i>	Specifies the name of the file that contains the server connection parameters. If not specified, the default file \$HOME/server.properties is used.
-H, --host <i>hostname</i>	Specifies the hostname or IP address of the WLS server.
-P, --port <i>port</i>	Specifies the port number to connect.
-u, --user <i>username</i>	Specifies the user name.
-s, --serv <i>server_name</i>	Specifies the name of the server on which ESSAPP is running.
-h	Displays the command usage.

3.6.8.3 Associated Files

The following file is used with this command:

`server.properties`: Contains server connection information

3.6.8.4 Exit Values

The command exits with the following possible values:

- 0: Success
- -1: Error

3.6.8.5 Examples

The following values apply to all of the examples:

`HOSTING_APP = ESS_NATIVE_HOSTING_APP_LOGICAL_NAME`

`PREDEF_SCHEDULE = /oracle/apps/ess/demo/seeded/MyPredefinedSch`

`SCHEDULE_NAME = MyWlstSchedule`

`SCH_DESC = Description for WLST test schedule`

and server connection properties are defined in the `~/server.properties` file.

- Show all schedule definitions present in namespace of `HOSTING_APP`:
`essManageSchedule.sh HOSTING_APP -S`
- Show details of the schedule definition `PREDEF_SCHEDULE` of `HOSTING_APP`:
`essManageSchedule.sh HOSTING_APP -S -n PREDEF_SCHEDULE`
- Create a new recurring schedule definition named `/oracle/apps/ess/custom/MyWlstSchedule` in the namespace of `HOSTING_APP` with the description `SCH_DESC`. The recurrence is: Occur every 2 minutes, starting at 21:10:30 on 03 May 2012, for 3 iterations:
`essManageSchedule.sh HOSTING_APP -C -n SCHEDULE_NAME -d SCH_DESC -F MINUTE -I 2 -c 3 -b 21:10:30:03:05:12`
- Create a new recurring schedule definition named `/oracle/apps/ess/custom/MyWlstSchedule` in the namespace of `HOSTING_APP`. The recurrence is: Occur every 2 months, on Thursday of the 5th week (if applicable), starting at 21:10:30 on 03 May 2012, and running until 21:10:30 on 04 May 2013:
`essManageSchedule.sh HOSTING_APP -C -n SCHEDULE_NAME -F MONTH -I 2 -b 21:10:30:03:05:12 -w 5 -x 4 -e 21:10:30:04:05:13`
- Create a new recurring schedule definition named `/oracle/apps/ess/custom/MyWlstSchedule` in the namespace of `HOSTING_APP`. The recurrence is: Occur every 3 years, on the last day of May, starting at 21:10:30 on 03 May 2012, for 5 iterations:
`essManageSchedule.sh HOSTING_APP -C -n SCHEDULE_NAME -F YEAR -I 3 -b 21:10:30:03:05:12 -m 5 -y LAST -c 5`
- Create a new recurring schedule definition named `/oracle/apps/ess/custom/MyWlstSchedule` in the namespace of `HOSTING_APP`. The recurrence is: Occur every hour, starting from the current time:
`essManageSchedule.sh HOSTING_APP -C -n SCHEDULE_NAME -F HOUR`

- Update the schedule definition `/oracle/apps/ess/custom+SCHEDULE_NAME` with the new description `SCH_DESC`. Recurrence remains unchanged:

```
essManageSchedule.sh HOSTING_APP -U -n SCHEDULE_NAME -d SCH_DESC
```

- Update the schedule definition `/oracle/apps/ess/custom+SCHEDULE_NAME` with the following new recurrence - occur every minute:

```
essManageSchedule.sh HOSTING_APP -U -n SCHEDULE_NAME -F MINUTE
```

- Delete the schedule definition `/oracle/apps/ess/custom+SCHEDULE_NAME` in `HOSTING_APP`:

```
essManageSchedule.sh HOSTING_APP -D -n SCHEDULE_NAME
```

3.6.8.6 Related Native Command

[Manage Oracle Enterprise Scheduler Schedule Definitions](#)

3.6.9 essBatchDeleteRequests

Submits a request for a batch delete job. User can either use a pre-defined schedule (in MDS) or specify a one-time execution time. If neither of these is specified, the request starts immediately.

The request parameters are used to specify the delete criteria. When the batch delete job runs, it determines which completed absolute parent and instance parent requests satisfy the delete criteria specified for that batch job request and deletes the request hierarchy for those requests.

For detailed list and description of valid batch delete parameters, see [Batch Delete Parameters](#).

The server connection parameters are specified using a file as described in [Oracle Enterprise Scheduler Convenience Scripts](#). However, the default values can be overridden by explicitly specifying them on the command line or specifying an alternate file using the `-f` option. The admin server password for WLS has to be provided in a file or entered interactively.

3.6.9.1 Syntax

```
essBatchDeleteRequests.sh [-d desc] [-S schId] [-b beginTime] [-e endTime]
[-f filename] [-H hostname] [-P port] [-u user] [-s server_name] [-h]
[prop=value ...]
```

3.6.9.2 Options

The following table lists and describes the command options.

Option	Description
<code>-d, --desc desc</code>	Specifies a description for the batch delete request.
<code>-S, --schId schId</code>	Specifies the fully qualified name of the schedule to use. The default package (used if the package is not specified) is <code>/oracle/as/ess/essapp/custom/</code> .

Option	Description
-b, --begin <i>beginTime</i>	Specifies the time at which the request is to be started. Format: HH:MM:SS:DD:MM:YYYY
-e, --end <i>endTime</i>	Specifies the time past which the request should not run. Format: HH:MM:SS:DD:MM:YYYY .
<i>prop=value</i>	Specifies other request parameters/properties specified as: <i>prop1=val1 prop2=val2</i> and so on. See Batch Delete Parameters for a list of the parameters.
-f, --file <i>filename</i>	Specifies the name of the file that contains the server connection parameters. If not specified, the default file <code>\$HOME/server.properties</code> is used.
-H, --host <i>hostname</i>	Specifies the hostname or IP address of the WLS server.
-P, --port <i>port</i>	Specifies the port number to connect.
-u, --user <i>username</i>	Specifies the user name.
-s, --serv <i>server_name</i>	Specifies the name of the server on which ESSAPP is running.
-h	Displays the command usage.

3.6.9.3 Associated Files

The following file is used with this command:

`server.properties`: Contains server connection information

3.6.9.4 Exit Values

The command exits with the following possible values:

- 0: Success
- -1: Error

3.6.9.5 Examples

The following examples assume that server connection properties are defined in the `~/server.properties` file.

- Delete all purgeable requests.:
`essBatchDeleteRequests.sh`
- Delete all purgeable requests in the application
`ESS_NATIVE_HOSTING_APP_LOGICAL_NAME`:
`essBatchDeleteRequests.sh --desc "My purge for ESS NativeApp"`
`CriteriaApplication=ESS_NATIVE_HOSTING_APP_LOGICAL_NAME`
- Delete all purgeable requests in application
`ESS_NATIVE_HOSTING_APP_LOGICAL_NAME`, for which the job definition is
`JobDefinition://oracle/apps/ess/custom/MyDef:`

```
essBatchDeleteRequests.sh --desc "My purge for ESS NativeApp"  
CriteriaApplication=ESS_NATIVE_HOSTING_APP_LOGICAL_NAMESCriteriaJobDefn=JobDefini  
tion://oracle/apps/ess/custom/MyDef
```

- **Submit batch delete job request using the schedule** /oracle/as/ess/essapp/
custom/WeeklySch:

```
essBatchDeleteRequests.sh --desc "Purge using WeekSch" --schId /oracle/as/ess/  
essapp/custom/WeeklySch
```

- **Delete all purgeable requests submitted by the user** weblogic:

```
essBatchDeleteRequests.sh CriteriaSubmitUser=weblogic
```

- **Delete at most ten purgeable requests:**

```
essBatchDeleteRequests.sh CriteriaProcessLimit=10
```

- **Delete purgeable requests whose job execution type is** JAVA_TYPE:

```
essBatchDeleteRequests.sh CriteriaJobExecType=JAVA_TYPE
```

- **Delete purgeable requests that completed at five or more days ago:**

```
essBatchDeleteRequests.sh CriteriaMinimumAge=5
```

- **Delete purgeable requests, but retain requests that completed successfully in the
past five days:**

```
essBatchDeleteRequests.sh CriteriaRetentionAgeSuccess=5
```

- **Delete purgeable requests that belong to the product named "DemoProduct":**

```
essBatchDeleteRequests.sh CriteriaProduct=DemoProduct
```

3.6.9.6 Related Native Command

[The batchDeleteSchedulerRequest Command](#)

4

Oracle Managed File Transfer Custom WLST Commands

This chapter summarizes WLST (Oracle WebLogic Scripting Tool) commands that perform Oracle Managed File Transfer (MFT) operations. This chapter includes the following sections:

- [Overview of MFT WLST Command Categories](#)
- [MFT Artifact Management Commands](#)
- [MFT Metadata Commands](#)
- [MFT Key Management Commands](#)
- [MFT Deployment History Commands](#)
- [MFT Transfer Management Commands](#)
- [MFT Embedded Server Commands](#)
- [MFT Callout Commands](#)
- [MFT Event Notification Commands](#)
- [MFT Archive and Restore Commands](#)
- [MFT Purge Commands](#)
- [Setting System MBean Properties for MFT WLST Commands](#)

For general information about WLST, see the *WLST Command Reference for WebLogic Server*.

For general information about MFT, see *Using Oracle Managed File Transfer*.

4.1 Overview of MFT WLST Command Categories

MFT WLST commands are divided into the following categories.

Table 4-1 MFT WLST Command Categories

Command Category	Description
MFT Artifact Management Commands	Perform these operations on source, transfer, and target artifacts: enable, disable, deploy, undeploy, delete, export, check existence in the Metadata Store (MDS).
MFT Metadata Commands	Export, import, and reset MFT metadata.
MFT Key Management Commands	Generate, import, export, delete, list, and update SSL, SSH, and PGP keys in the MFT keystore.
MFT Deployment History Commands	View the deployment history of source, transfer, and target artifacts.

Table 4-1 (Cont.) MFT WLST Command Categories

Command Category	Description
MFT Transfer Management Commands	Pause, resume, resubmit, or display information about transfer instances.
MFT Embedded Server Commands	Start, stop, and change ports of embedded FTP and sFTP servers.
MFT Callout Commands	Create, delete, list, and update custom callouts.
MFT Event Notification Commands	<ul style="list-style-type: none"> • Create, delete, and list contacts • Add and remove contacts from event notifications • Enable and disable event notifications
MFT Archive and Restore Commands	Archive transfer instances or file system data based on criteria such as status and date ranges.
MFT Purge Commands	Purge transfer instances or file system data based on criteria such as status and date ranges.

4.2 MFT Artifact Management Commands

Use the MFT WLST Artifact Management commands, listed in [Table 4-2](#), to perform operations on source, transfer, and target artifacts.

Table 4-2 MFT Artifact Management WLST Commands

Use this command...	To...	Use with WLST
bulkDeployArtifact	Deploys multiple source, transfer, or target artifacts.	Online or Offline
createArtifacts or <code>crtAF</code>	Create Artifacts from an input xml file containing artifact definition.	Online or Offline
deleteArtifact or <code>delAF</code>	Delete a source, transfer, or target artifact.	Online or Offline
deleteArtifactDeployment or <code>delDepAF</code>	Delete an undeployed source, transfer, or target artifact.	Online or Offline
deployArtifact or <code>depAF</code>	Deploy a source, transfer, or target artifact.	Online or Offline
disableArtifact or <code>disAF</code>	Disable a deployed source, transfer, or target artifact.	Online or Offline
enableArtifact or <code>enAF</code>	Enable a deployed source, transfer, or target artifact.	Online or Offline
exportDeployedArtifact or <code>expDepAF</code>	Export a deployed source, transfer, or target artifact to a ZIP file.	Online or Offline
isArtifactInMDS or <code>isAFinMDS</code>	Check whether a source, transfer, or target artifact exists in the MDS (Metadata Store).	Online or Offline
undeployArtifact or <code>undepAF</code>	Undeploy a source, transfer, or target artifact.	Online or Offline

4.2.1 bulkDeployArtifact

Command Category: MFT Artifact Management Commands

Use with WLST: Online or Offline

4.2.1.1 Description

Bulk deploys multiple source, transfer, or target artifacts. In the event of an unsupported operation, the command returns a `WLSTException`.

4.2.1.2 Syntax

```
bulkDeployArtifact(artifact_type, artifact_names, comment)
```

Argument	Definition
<i>artifact_type</i>	Artifact type: SOURCE, TRANSFER, or TARGET.
<i>artifact_names</i>	Comma-separated artifact names, or * for all.
<i>comment</i>	Text string describing the artifacts or the reason for deploying them.

4.2.1.3 Example

The following example deploys two SOURCE artifacts and provides a comment.

```
bulkDeployArtifact('SOURCE', 'order-file-src,chk-inv-src', 'retrieves new purchase orders, checks inventory')
```

The following example deploys all TRANSFER artifacts and provides a comment.

```
bulkDeployArtifact('TRANSFER', '*', 'deploying all transfers')
```

4.2.2 createArtifacts

Command Category: MFT Artifact Management Commands

Description

Use with WLST: Online or Offline

Creates Artifacts from an input xml file containing artifact definition. Shortcut for this command is `crtAF`

Syntax

```
createArtifacts('<xmlFilePath>', previewMode, updateIfExists)
```

Argument	Definition
<i>xmlFilePath</i>	XML file location containing artifact definition.
<i>previewMode</i>	Lists the modified artifacts.
<i>updateIfExists</i>	Updates the existing artifact, if artifact does not exist, it will fail.

Example 4-1 Example of createArtifacts command

The following example creates an artifact from the input file called `file.xml`.

```
createArtifacts('/scratch/file.xml', previewMode, updateIfExists)
```

4.2.3 deleteArtifact

Command Category: MFT Artifact Management Commands

Use with WLST: Online or Offline

4.2.3.1 Description

Deletes a source, transfer, or target artifact. In the event of an unsupported operation, the command returns a `WLSTException`.

4.2.3.2 Syntax

```
deleteArtifact(artifact_type, artifact_name)
```

```
delAF(artifact_type, artifact_name)
```

Argument	Definition
<i>artifact_type</i>	Artifact type: SOURCE, TRANSFER, or TARGET.
<i>artifact_name</i>	Artifact name.

4.2.3.3 Example

The following example deletes a SOURCE artifact called `order-file-src`.

```
deleteArtifact('SOURCE', 'order-file-src')
```

4.2.4 deleteArtifactDeployment

Command Category: MFT Artifact Management Commands

Use with WLST: Online or Offline

4.2.4.1 Description

Deletes an undeployed source, transfer, or target artifact. In the event of an unsupported operation, the command returns a `WLSTException`.

4.2.4.2 Syntax

```
deleteArtifactDeployment(artifact_type, artifact_name, label)
```

```
delDepAF(artifact_type, artifact_name, label)
```

Argument	Definition
<i>artifact_type</i>	Artifact type: SOURCE, TRANSFER, or TARGET.
<i>artifact_name</i>	Artifact name.
<i>label</i>	Artifact label in the MDS (Metadata Store). Use Show Deployment Details on the Deployment tab to view this label.

4.2.4.3 Example

The following example deletes a SOURCE artifact called `order-file-src` with the label `soa_mft-2012-12-07 22:24:09.383`.

```
deleteArtifactDeployment('SOURCE', 'order-file-src', 'soa_mft-2012-12-07  
22:24:09.383')
```

4.2.5 deployArtifact

Command Category: MFT Artifact Management Commands

Use with WLST: Online or Offline

4.2.5.1 Description

Deploys a source, transfer, or target artifact. In the event of an unsupported operation, the command returns a `WLSTException`.

4.2.5.2 Syntax

```
deployArtifact(artifact_type, artifact_name, comment)
```

```
depAF(artifact_type, artifact_name, comment)
```

Argument	Definition
<i>artifact_type</i>	Artifact type: SOURCE, TRANSFER, or TARGET.
<i>artifact_name</i>	Artifact name.

Argument	Definition
<i>comment</i>	Text string describing the artifact or the reason for deploying it.

4.2.5.3 Example

The following example deploys a SOURCE artifact called `order-file-src` and provides a comment about the artifact.

```
deployArtifact('SOURCE', 'order-file-src', 'retrieves new purchase orders')
```

4.2.6 disableArtifact

Command Category: MFT Artifact Management Commands

Use with WLST: Online or Offline

4.2.6.1 Description

Disables a deployed and previously enabled source, transfer, or target artifact. In the event of an unsupported operation, the command returns a `WLSTException`.

4.2.6.2 Syntax

```
disableArtifact(artifact_type, artifact_name, comment)
```

```
disAF(artifact_type, artifact_name, comment)
```

Argument	Definition
<i>artifact_type</i>	Artifact type: SOURCE, TRANSFER, or TARGET.
<i>artifact_name</i>	Artifact name.
<i>comment</i>	Text string describing the artifact or the reason for disabling it.

4.2.6.3 Example

The following example disables a SOURCE artifact called `order-file-src` and provides a comment about the reason.

```
disableArtifact('SOURCE', 'order-file-src', 'order server being upgraded')
```

4.2.7 enableArtifact

Command Category: MFT Artifact Management Commands

Use with WLST: Online or Offline

4.2.7.1 Description

Enables a deployed and previously disabled source, transfer, or target artifact. In the event of an unsupported operation, the command returns a `WLSTException`.

4.2.7.2 Syntax

```
enableArtifact(artifact_type, artifact_name, enableAssociatedArtifacts, comments)
```

```
enAF(artifact_type, artifact_name, enableAssociatedArtifacts, comments)
```

Argument	Definition
<i>artifact_type</i>	Artifact type: SOURCE, TRANSFER, or TARGET.
<i>artifact_name</i>	Artifact name.
<i>enableAssociatedArtifacts</i> (optional, default true)	Indicates whether to enable the associate artifacts or not.
<i>comment</i>	Text string describing the artifact or the reason for enabling it.

4.2.7.3 Example

The following example enables a SOURCE artifact called `order-file-src` and provides a comment about the reason.

```
enableArtifact('SOURCE', 'order-file-src', 'order server upgrade complete')
```

4.2.8 exportDeployedArtifact

Command Category: MFT Artifact Management Commands

Use with WLST: Online or Offline

4.2.8.1 Description

Exports a deployed source, transfer, or target artifact to a ZIP file. In the event of an unsupported operation, the command returns a `WLSTException`.

4.2.8.2 Syntax

```
exportDeployedArtifact(<ArtifactType>', '<ArtifactName>', <Label>', '<ArchiveFilePath>',  
generateConfigPlan, longFormat)
```

```
expDepAF(artifact_type, artifact_name, label, archive_file_path,  
label, generateConfigPlan, longFormat)
```

Argument	Definition
<i>artifact_type</i>	Artifact type: SOURCE, TRANSFER, or TARGET.
<i>artifact_name</i>	Name of the artifact to exported.
<i>label</i>	Artifact label in the MDS (Metadata Store). Use Show Deployment Details on the Deployment tab to view this label.
<i>archive_file_path</i>	Full path to the ZIP file to which to export. If you are connecting to WLST remotely, the ZIP file is created on the remote server.
<i>generateConfigPlan</i> (optional, default false)	Indicates whether to generate the mftConfig XML. Config plan will be generated in the same folder where archive file will be generated.
<i>longFormat</i> (optional, default false)	If passed true, most of the attributes will be included in the config plan, else only the key attributes will be listed in the config plan XML.

4.2.8.3 Example

The following example exports a SOURCE artifact called `order-file-src` with the label `soa_mft-2012-12-07 22:24:09.383` to `/export/order-file-src.zip`.

```
exportDeployedArtifact('SOURCE', 'order-file-src', 'soa_mft-2012-12-07
22:24:09.383', '/export/order-file-src.zip')
```

4.2.9 isArtifactInMDS

Command Category: MFT Artifact Management Commands

Use with WLST: Online or Offline

4.2.9.1 Description

Checks whether a source, transfer, or target artifact exists in the MDS (Metadata Store) and returns TRUE or FALSE. In the event of an unsupported operation, the command returns a `WLSTException`.

4.2.9.2 Syntax

```
isArtifactInMDS(artifact_type, artifact_name)
```

```
isAFinMDS(artifact_type, artifact_name)
```

Argument	Definition
<i>artifact_type</i>	Artifact type: SOURCE, TRANSFER, or TARGET.
<i>artifact_name</i>	Artifact name.

4.2.9.3 Example

The following example checks whether a `SOURCE` artifact called `order-file-src` exists in the MDS.

```
isArtifactInMDS('SOURCE', 'order-file-src')
```

4.2.10 undeployArtifact

Command Category: MFT Artifact Management Commands

Use with WLST: Online or Offline

4.2.10.1 Description

Undeploys a source, transfer, or target artifact without deleting it from the configuration. In the event of an unsupported operation, the command returns a `WLSTException`.

4.2.10.2 Syntax

```
undeployArtifact(artifact_type, artifact_name, comment)
```

```
undepAF(artifact_type, artifact_name, comment)
```

Argument	Definition
<i>artifact_type</i>	Artifact type: SOURCE, TRANSFER, or TARGET.
<i>artifact_name</i>	Artifact name.
<i>comment</i>	Text string describing the artifact or the reason for undeploying it.

4.2.10.3 Example

The following example undeploys a `SOURCE` artifact called `order-file-src` and provides a comment about the reason.

```
undeployArtifact('SOURCE', 'order-file-src', 'upgrading artifact')
```

4.3 MFT Metadata Commands

Use the MFT WLST Metadata commands, listed in [Table 4-3](#), to perform metadata operations.

Table 4-3 MFT Metadata WLST Commands

Use this command...	To...	Use with WLST
<code>exportMftMetadata</code> or <code>expMD</code>	Export the entire MFT configuration, excluding passwords, to a ZIP file.	Online or Offline
<code>exportTransferMetadata</code> or <code>expXfrMD</code>	Export a transfer artifact and related metadata to a ZIP file.	Online or Offline
<code>importMftMetadata</code> or <code>impMD</code>	Import a previously exported MFT configuration from a ZIP file.	Online or Offline
<code>resetMetadata</code> or <code>resMD</code>	Reset the MFT configuration, deleting all artifacts and resetting all administrative settings to their defaults, while optionally preserving user preferences.	Online or Offline

4.3.1 exportMftMetadata

Command Category: MFT Metadata Commands

Use with WLST: Online or Offline

4.3.1.1 Description

Exports the entire MFT configuration, excluding passwords, to a ZIP file. In the event of an unsupported operation, the command returns a `WLSTException`.

4.3.1.2 Syntax

```
exportMftMetadata(ArchiveFile, generateConfigPlan, longFormat)
```

```
expMD(ArchiveFile, generateConfigPlan, longFormat)
```

Argument	Definition
<code>archiveFile</code>	Full path to the ZIP file to which to export.
<code>generateConfigPlan</code>	It is optional, default is false. It indicates whether to generate the <code>mftConfig</code> XML. Config plan is generated in same folder where archive file is generated.
<code>longFormat</code>	It is optional, default is false. If true, most of the attributes will be included in the config plan xml; otherwise, only the key attributes will be listed in the config plan XML.

4.3.1.3 Example

The following example exports the MFT configuration to `/export/mft-config-dec-2012.zip`.

```
exportMftMetadata('/export/mft-config-dec-2012.zip')
```

4.3.2 exportTransferMetadata

Command Category: MFT Metadata Commands

Use with WLST: Online or Offline

4.3.2.1 Description

Exports a transfer artifact and related metadata to a ZIP file. In the event of an unsupported operation, the command returns a `WLSTException`.

4.3.2.2 Syntax

```
exportTransferMetadata(archiveFile, transfer_name, generateConfigPlan, longFormat)
```

```
expXfrMD(archiveFile, transfer_name, generateConfigPlan, longFormat)
```

Argument	Definition
<code>archiveFile</code>	Full path to the ZIP file to which to export.
<code>transfer_name</code>	Transfer artifact name.
<code>generateConfigPlan</code>	It is optional, default value is false. It indicates whether to generate the mftConfig XML. Config plan is generated in same folder where archive file is generated.
<code>longFormatarchiveFile</code>	It is optional, default value is false. If true, most of the attributes will be included in the config plan xml; otherwise, only the key attributes will be listed in the config plan XML.

4.3.2.3 Example

The following example exports a transfer artifact named `order-xfr` and its metadata to `/export/order-xfr.zip`.

```
exportTransferMetadata('/export/order-xfr.zip', 'order-xfr')
```

4.3.3 importMftMetadata

Command Category: MFT Metadata Commands

Use with WLST: Online or Offline

4.3.3.1 Description

Imports a previously exported MFT configuration from a ZIP file. In the event of an unsupported operation, the command returns a `WLSTException`.

4.3.3.2 Syntax

```
importMftMetadata(ArchiveFile, MftConfigPlanXML, previewMode)
```

```
impMD(ArchiveFile, MftConfigPlanXML, previewMode)
```

Argument	Definition
<i>ArchiveFile</i>	File location for importing metadata.
<i>MftConfigPlanXML</i>	File location for mftConfig XML
(optional)	
<i>previewMode</i>	It is optional, default value is false. If passed true, no metadata will be imported. It can be used just to get info like which all attributes will be changed because of config plan.

4.3.3.3 Example

The following example imports the MFT configuration from `/export/mft-config-dec-2012.zip`.

```
importMftMetadata('/export/mft-config-dec-2012.zip')
```

4.3.4 resetMetadata

Command Category: MFT Metadata Commands

Use with WLST: Online or Offline

4.3.4.1 Description

Resets the MFT configuration, deleting all artifacts and resetting all administrative settings to their defaults, while optionally preserving user preferences. In the event of an unsupported operation, the command returns a `WLSTException`.

4.3.4.2 Syntax

```
resetMetadata(preserve_preferences)
```

```
resMD(preserve_preferences)
```

Argument	Definition
<i>preserve_preferences</i>	If TRUE, preserves user preferences. If FALSE, resets user preferences along with the rest of the configuration.

4.3.4.3 Example

The following example resets the MFT configuration but preserves user preferences.

```
resetMetadata('TRUE')
```

4.4 MFT Key Management Commands

Use the MFT WLST Key Management commands, listed in [Table 4-4](#), to manage SSL, SSH, and PGP keys in the MFT keystore. For additional information, see "Keystore Management" in *Using Oracle Managed File Transfer*.

Table 4-4 MFT Key Management WLST Commands

Use this command...	To...	Use with WLST
<code>deleteCSFKey</code> or <code>delKey</code>	Delete a key alias from the MFT keystore.	Online or Offline
<code>exportCSFKey</code> or <code>expKey</code>	Export keys from the MFT keystore to a key file.	Online or Offline
<code>generateKeys</code> or <code>genKeys</code>	Generate keys and save them to one or more key files.	Online or Offline
<code>importCSFKey</code> or <code>impKey</code>	Import a key to the MFT keystore from a key file.	Online or Offline
<code>listCSFKeyAliases</code> or <code>lsKeyAliases</code>	List key aliases in the MFT keystore.	Online or Offline
<code>updateCSFKey</code> or <code>updKey</code>	Delete a key alias from the MFT keystore and generate a new key file.	Online or Offline

4.4.1 deleteCSFKey

Command Category: MFT Key Management Commands

Use with WLST: Online or Offline

4.4.1.1 Description

Deletes a key alias from the MFT keystore. In the event of an unsupported operation, the command returns a `WLSTException`.

4.4.1.2 Syntax

```
deleteCSFKey(key_format, key_type, alias)
```

```
delKey(key_format, key_type, alias)
```

Argument	Definition
<code>key_format</code>	Key format: SSH or PGP.
<code>key_type</code>	Key format: PRIVATE or PUBLIC.
<code>alias</code>	Key alias.

4.4.1.3 Example

The following example deletes a PRIVATE SSH key with the alias my-alias.

```
deleteCSFKey('SSH', 'PRIVATE', 'my-alias')
```

4.4.2 exportCSFKey

Command Category: MFT Key Management Commands

Use with WLST: Online or Offline

4.4.2.1 Description

Exports keys from the MFT keystore to a key file. In the event of an unsupported operation, the command returns a `WLSTException`.

4.4.2.2 Syntax

```
exportCSFKey(key_format, key_type, key_file_path)
```

```
expKey(key_format, key_type, key_file_path)
```

Argument	Definition
<i>key_format</i>	Key format: SSH or PGP.
<i>key_type</i>	Key format: PRIVATE or PUBLIC.
<i>key_file_path</i>	Full path to the key file to which to export.

4.4.2.3 Example

The following example exports PRIVATE SSH keys to the file `/export/ssh/my_private_keys.ppk`.

```
exportCSFKey('SSH', 'PRIVATE', '/export/ssh/my_private_keys.ppk')
```

4.4.3 generateKeys

Command Category: MFT Key Management Commands

Use with WLST: Online or Offline

4.4.3.1 Description

Generates keys and saves them to one or more key files. The key type is RSA and the key size is 1024 bits. In the event of an unsupported operation, the command returns a `WLSTException`.

4.4.3.2 Syntax

```
generateKeys (<KeyFormat>,<Password>,<FileLocation>,<KeyIdentity>,<KeySize>)
```

```
genKeys(<KeyFormat>,<Password>,<FileLocation>,<KeyIdentity>,<KeySize>)
```

Argument	Definition
<i>KeyFormat</i>	Key format: SSH or PGP.
<i>Password</i>	Password for the private key.
<i>FileLocation</i>	Directory where the generated key(s) will be placed. For SSH the location should also contain the key file name.
<i>KeyIdentity</i>	Identity String to be used with PGP keys. For SSH key this will be ignored. This is an optional property. If not specified, it defaults to "mft pgp key - KeyGen Util" .
<i>KeySize</i>	Size of the key to be generated. If not provided, then by default keys with size 1024 is generated. Min size = 1024 and Max size = 8192 and the restrictions are applicable for both SSH and PGP keys.

4.4.3.3 Example

The following example exports SSH keys without password protection to the file /export/ssh/ssh-pvt-keys.ppk.

```
generateKeys('SSH', '', '/export/ssh/ssh-pvt-keys.ppk')
```

The following example exports PGP keys with password protection to the directory /export/pgp.

```
generateKeys('PGP', 'P@s$W0rd', '/export/pgp')
```

4.4.4 importCSFKey

Command Category: MFT Key Management Commands

Use with WLST: Online or Offline

4.4.4.1 Description

Imports a key to the MFT keystore from a key file and creates an alias. In the event of an unsupported operation, the command returns a `WLSTException`.

4.4.4.2 Syntax

```
importCSFKey(key_format, key_type, alias, key_file_path)
```

```
impKey(key_format, key_type, alias, key_file_path)
```

Argument	Definition
<i>key_format</i>	Key format: SSH or PGP.
<i>key_type</i>	Key format: PRIVATE or PUBLIC.
<i>alias</i>	Key alias.
<i>key_file_path</i>	Full path to the key file from which to import.

4.4.4.3 Example

The following example imports a PRIVATE SSH key with the alias `my-alias` from the file `/export/ssh/my_private_keys.ppk`.

```
importCSFKey('SSH', 'PRIVATE', 'my-alias', '/export/ssh/my_private_keys.ppk')
```

The following example imports a PUBLIC PGP key with the alias `mftpgppub` from the file `/export/pgp/pub.asc`.

```
importCSFKey('PGP', 'PUBLIC', 'mftpgppub', '/export/pgp/pub.asc')
```

The following example imports a PRIVATE PGP key with the alias `mftpgppri` from the file `/export/pgp/secret.asc`.

```
importCSFKey('PGP', 'PRIVATE', 'mftpgppri', '/export/pgp/secret.asc')
```

4.4.5 listCSFKeyAliases

Command Category: MFT Key Management Commands

Use with WLST: Online or Offline

4.4.5.1 Description

Lists key aliases in the MFT keystore. In the event of an unsupported operation, the command returns a `WLSTException`.

4.4.5.2 Syntax

```
listCSFKeyAliases(key_format, key_type, alias)
```

```
lsKeyAliases(key_format, key_type, alias)
```

Argument	Definition
<i>key_format</i>	Key format: SSH or PGP.
<i>key_type</i>	Key format: PRIVATE or PUBLIC.

Argument	Definition
<i>alias</i> (optional)	If specified, it provides the details of the given key stored with the given alias.

4.4.5.3 Example

The following example lists PRIVATE SSH keys.

```
listCSFKeyAliases('SSH', 'PRIVATE')
```

4.4.6 updateCSFKey

Command Category: MFT Key Management Commands

Use with WLST: Online or Offline

4.4.6.1 Description

Deletes a key alias from the MFT keystore and generates a new key file. In the event of an unsupported operation, the command returns a `WLSTException`.

4.4.6.2 Syntax

```
updateCSFKey(key_format, key_type, alias, key_file_path)
```

```
updKey(key_format, key_type, alias, key_file_path)
```

Argument	Definition
<i>key_format</i>	Key format: SSH or PGP.
<i>key_type</i>	Key format: PRIVATE or PUBLIC.
<i>alias</i>	Key alias.
<i>key_file_path</i>	Full path to the key file to generate.

4.4.6.3 Example

The following example deletes a PRIVATE SSH key with the alias `my-alias` and generates the key file `/export/ssh/my-private-key.ppk`.

```
updateCSFKey('SSH', 'PRIVATE', 'my-alias', '/export/ssh/my-private-key.ppk')
```

4.5 MFT Deployment History Commands

Use the MFT WLST Deployment History commands, listed in [Table 4-5](#), to view the deployment history of source, transfer, and target artifacts.

Table 4-5 MFT Deployment History WLST Commands

Use this command...	To...	Use with WLST
getSourceDeploymentHistory or <code>getSrcDH</code>	Return the deployment history of a source artifact.	Online or Offline
getTargetDeploymentHistory or <code>getTrgtDH</code>	Return the deployment history of a target artifact.	Online or Offline
getTransferDeploymentHistory or <code>getXfrDH</code>	Return the deployment history of a transfer artifact.	Online or Offline

4.5.1 getSourceDeploymentHistory

Command Category: MFT Deployment History Commands

Use with WLST: Online or Offline

4.5.1.1 Description

Returns the deployment history of a source artifact. In the event of an unsupported operation, the command returns a `WLSTException`.

4.5.1.2 Syntax

```
getSourceDeploymentHistory(source_name)
```

```
getSrcDH(source_name)
```

Argument	Definition
<i>source_name</i>	Source artifact name.

4.5.1.3 Example

The following example returns the deployment history of a source artifact named `order-file-src`.

```
getSourceDeploymentHistory('order-file-src')
```

4.5.2 getTargetDeploymentHistory

Command Category: MFT Deployment History Commands

Use with WLST: Online or Offline

4.5.2.1 Description

Returns the deployment history of a target artifact. In the event of an unsupported operation, the command returns a `WLSTException`.

4.5.2.2 Syntax

```
getTargetDeploymentHistory(target_name)
```

```
getTrgtDH(target_name)
```

Argument	Definition
<i>target_name</i>	Target artifact name.

4.5.2.3 Example

The following example returns the deployment history of a target artifact named `order-file-tgt`.

```
getTargetDeploymentHistory('order-file-tgt')
```

4.5.3 getTransferDeploymentHistory

Command Category: MFT Deployment History Commands

Use with WLST: Online or Offline

4.5.3.1 Description

Returns the deployment history of a transfer artifact. In the event of an unsupported operation, the command returns a `WLSTException`.

4.5.3.2 Syntax

```
getTransferDeploymentHistory(transfer_name)
```

```
getXfrDH(transfer_name)
```

Argument	Definition
<i>transfer_name</i>	Transfer artifact name.

4.5.3.3 Example

The following example returns the deployment history of a transfer artifact named `order-xfr`.

```
getTransferDeploymentHistory('order-xfr')
```

4.6 MFT Transfer Management Commands

Use the MFT WLST Transfer Management commands, listed in [Table 4-6](#), to manage transfer instances.

Table 4-6 MFT Transfer Management WLST Commands

Use this command...	To...	Use with WLST
<code>getTransferInfo</code> or <code>getXfrInfo</code>	Return information about a transfer artifact.	Online or Offline
<code>pauseTransfer</code> or <code>pauseXfr</code>	Pause an in-progress transfer.	Online or Offline
<code>resubmit</code> or <code>resub</code>	Resubmit a failed transfer.	Online or Offline
<code>resumeTransfer</code> or <code>resXfr</code>	Resume a paused transfer.	Online or Offline

4.6.1 getTransferInfo

Command Category: MFT Transfer Management Commands

Use with WLST: Online or Offline

4.6.1.1 Description

Returns information about a transfer artifact. In the event of an unsupported operation, the command returns a `WLSTException`.

4.6.1.2 Syntax

```
getTransferInfo(transfer_name, label)
```

```
getXfrInfo(transfer_name, label)
```

Argument	Definition
<code>transfer_name</code>	Transfer artifact name.
<code>label</code>	Artifact label in the MDS (Metadata Store). Use Show Deployment Details on the Deployment tab to view this label.

4.6.1.3 Example

The following example returns information about a transfer artifact named `order-xfr` with the label `soa_mft-2012-12-07 22:28:17.392`.

```
getTransferInfo('order-xfr', 'soa_mft-2013-09-26 22:28:17.392')
ID                                     | NAME                               | LABEL
xfer_521a6788-4e24-4822-bb03-43a4a8eaa8ce | ftp-file-xfer                     | soa_mft-2013-09-26
16:00:09.803
```

4.6.2 pauseTransfer

Command Category: MFT Transfer Management Commands

Use with WLST: Online or Offline

4.6.2.1 Description

Pauses an in-progress transfer. In the event of an unsupported operation, the command returns a `WLSTException`.

4.6.2.2 Syntax

```
pauseTransfer(instance_id, comment)
```

```
pauseXfr(instance_id, comment)
```

Argument	Definition
<code>instance_id</code>	Target instance ID for the transfer. Open the Advanced section of the target report to view the instance ID. For information about the target report, see "Interpreting Source, Transfer, and Target Reports" in <i>Using Oracle Managed File Transfer</i> .
<code>comment</code>	Text string describing the reason for pausing.

4.6.2.3 Example

The following example pauses a transfer with the target instance ID 240C93AD-5401-483B-8182-274FA0705DF1 and provides a reason.

```
pauseTransfer('240C93AD-5401-483B-8182-274FA0705DF1', 'resume when less network traffic')
```

4.6.3 resubmit

Command Category: MFT Transfer Management Commands

Use with WLST: Online or Offline

4.6.3.1 Description

Resubmits a transfer. You can resubmit a successful or failed transfer. In the event of an unsupported operation, the command returns a `WLSTException`.

4.6.3.2 Syntax

```
resubmit(resubmit_type, instance_ids, comment)
```

```
resub(resubmit_type, instance_ids, comment)
```

Argument	Definition
<code>resubmit_type</code>	Resubmit type: SOURCE, TRANSFER_INSTANCE, TARGET, or TARGET_INSTANCE. For TARGET, the file is delivered without preprocessing. For TARGET_INSTANCE, any preprocessing is called again before delivery.

Argument	Definition
<i>instance_ids</i>	Comma-separated instance IDs. Open the Advanced section of the source, transfer, or target report to view the instance ID. For information about these reports, see "Interpreting Source, Transfer, and Target Reports" in <i>Using Oracle Managed File Transfer</i> .
<i>comment</i>	Text string describing the reason for resubmitting.

4.6.3.3 Example

The following example resubmits a SOURCE with the instance ID 3D48B12B-295A-4F52-A8EE-BD1CC1A20246 and provides a reason.

```
resubmit('SOURCE', '3D48B12B-295A-4F52-A8EE-BD1CC1A20246', 'trying again')
```

4.6.4 resumeTransfer

Command Category: MFT Transfer Management Commands

Use with WLST: Online or Offline

4.6.4.1 Description

Resumes a paused transfer. In the event of an unsupported operation, the command returns a `WLSTException`.

4.6.4.2 Syntax

```
resumeTransfer(instance_id, comment)
```

```
resXfr(instance_id, comment)
```

Argument	Definition
<i>instance_id</i>	Target instance ID for the transfer. Open the Advanced section of the target report to view the instance ID. For information about the target report, see "Interpreting Source, Transfer, and Target Reports" in <i>Using Oracle Managed File Transfer</i> .
<i>comment</i>	Text string describing the reason for resuming.

4.6.4.3 Example

The following example resumes a transfer with the target instance ID 240C93AD-5401-483B-8182-274FA0705DF1 and provides a reason.

```
resumeTransfer('240C93AD-5401-483B-8182-274FA0705DF1', 'less network traffic now')
```


4.7 MFT Embedded Server Commands

Use the MFT WLST Embedded Server commands, listed in [Table 4-7](#), to manage embedded FTP and sFTP servers.

Table 4-7 MFT Embedded Server WLST Commands

Use this command...	To...	Use with WLST
<code>ConfigureHomeDir</code> or <code>confHmDir</code>	Assigns a home directory on an embedded server.	Online or Offline
<code>grantPermissionToDirectory</code> or <code>grPermDir</code>	Grant permission to an embedded server directory.	Online or Offline
<code>listAllPermissions</code> or <code>lsPerms</code>	List all permissions available for a given principal and server type.	Online or Offline
<code>revokePermissionForDirectory</code> or <code>revPermDir</code>	Revoke permissions of an embedded server directory.	Online or Offline
<code>startEmbeddedServer</code> or <code>startES</code>	Start an embedded FTP or sFTP server that was stopped.	Online or Offline
<code>stopEmbeddedServer</code> or <code>stopES</code>	Stop an embedded FTP or sFTP server that is running.	Online or Offline
<code>updatePorts</code> or <code>updPorts</code>	Update the port for an embedded FTP or sFTP server.	Online or Offline

4.7.1 ConfigureHomeDir

Command Category: MFT Embedded Server Commands

Use with WLST: Online or Offline

4.7.1.1 Description

Assigns the specified directory to the user as home directory where that user is located on login to embedded servers.

4.7.1.2 Syntax

```
configureHomeDir(directory_path, user_name)
```

Argument	Definition
<code>directory_path</code>	Any valid path starting from the embedded server root.
<code>user_name</code>	A WebLogic user ID

4.7.1.3 Example

The following example assigns directory `dir1` as the home directory for a user named "user1".

```
configureHomeDir("/dir1","user1")
```

4.7.2 grantPermissionToDirectory

Command Category: MFT Embedded Server Commands

Use with WLST: Online or Offline

4.7.2.1 Description

Grant permission to an embedded server directory. Users and groups can be assigned a set of permissions to an existing directory on an embedded server.

4.7.2.2 Syntax

```
grantPermissionToDirectory(directory_path, principal_name, principal_type,
permissions, server_type, include_subfolder)
```

Argument	Definition
directory_path	Any valid path starting from the embedded server root
principal_name	An enterprise user, group or application role name.
principal_type	One of the following: <ul style="list-style-type: none"> • USER • GROUP • APPLICATION_ROLE
permissions	Comma separated list of the following permissions: <ul style="list-style-type: none"> • READ • WRITE • LIST • DELETE
server_type	FTP or SFTP
include_subfolder	true or false

4.7.2.3 Example

The following example assigns read, write, list and delete permissions to the directory `/orders` (and sub-directories) for the user "weblogic" on an FTP server:

```
grantPermissionToDirectory("/orders","weblogic","USER", "READ, WRITE, LIST,
DELETE","FTP",true)
```

4.7.3 listAllPermissions

Command Category: MFT Embedded Server Commands

Use with WLST: Online or Offline

4.7.3.1 Description

List all permissions available for a given principal and server type. The server type can be FTP or SFTP.

4.7.3.2 Syntax

```
listAllPermissions(principal_name, server_type)
```

Argument	Definition
<i>principal_name</i>	An enterprise user, group or application role name.
<i>server_type</i>	FTP or SFTP

4.7.3.3 Example

The following example lists all permissions for the principal named "weblogic" on an FTP server.

```
listAllPermissions("weblogic", "FTP")
```

4.7.4 revokePermissionForDirectory

Command Category: MFT Embedded Server Commands

Use with WLST: Online or Offline

4.7.4.1 Description

Revoke a set of permissions from an embedded server directory.

4.7.4.2 Syntax

```
revokePermissionForDirectory(directory_path, principal_name, principal_type,  
permissions, server_type, include_subfolder)
```

Argument	Definition
<i>directory_path</i>	Any valid path starting from the embedded server root
<i>principal_name</i>	An enterprise user, group or application role name.
<i>principal_type</i>	One of the following: <ul style="list-style-type: none"> • USER • GROUP • APPLICATION_ROLE
<i>permissions</i>	Comma separated list of the following permissions: <ul style="list-style-type: none"> • READ • WRITE • LIST • DELETE

Argument	Definition
<code>server_type</code>	FTP or SFTP
<code>include_subfolder</code>	true or false

4.7.4.3 Example

The following example revokes read, write, list and delete permissions from the directory `/orders` (and sub-directories) for the user "weblogic" on an FTP server:

```
grantPermissionToDirectory("/orders", "weblogic", "USER", "READ, WRITE, LIST,
DELETE", "FTP", true)
```

4.7.5 startEmbeddedServer

Command Category: MFT Embedded Server Commands

Use with WLST: Online or Offline

4.7.5.1 Description

Starts an embedded FTP, FTPS (FTP over SSL), or SFTP (SSH-FTP) server that was stopped. In the event of an unsupported operation, the command returns a `WLSTException`.

4.7.5.2 Syntax

```
startEmbeddedServer(server_type)
```

```
startES(server_type)
```

Argument	Definition
<code>server_type</code>	Embedded server type: FTP, FTPS, or SFTP.

4.7.5.3 Example

The following example starts the embedded FTP server.

```
startEmbeddedServer('FTP')
```

4.7.6 stopEmbeddedServer

Command Category: MFT Embedded Server Commands

Use with WLST: Online or Offline

4.7.6.1 Description

Stops an embedded FTP, FTPS (FTP over SSL), or SFTP (SSH-FTP) server that is running. In the event of an unsupported operation, the command returns a `WLSTException`.

4.7.6.2 Syntax

```
stopEmbeddedServer(server_type)
```

```
stopES(server_type)
```

Argument	Definition
<i>server_type</i>	Embedded server type: FTP, FTPS, or SFTP.

4.7.6.3 Example

The following example stops the embedded FTP server.

```
stopEmbeddedServer('FTP')
```

4.7.7 updatePorts

Command Category: MFT Embedded Server Commands

Use with WLST: Online or Offline

4.7.7.1 Description

Updates the port for an embedded FTP, FTPS (FTP over SSL), or sFTP (SSH-FTP) server, which is a service of an Oracle WebLogic Server managed server dedicated to MFT. In the event of an unsupported operation, the command returns a `WLSTException`.

4.7.7.2 Syntax

```
updatePorts(server_instance, service, port)
```

```
updPorts(server_instance, service, port)
```

Argument	Definition
<i>server_instance</i>	Name of Oracle WebLogic Server managed server dedicated to MFT.
<i>service</i>	Service (embedded server type): FTP, FTPS, or SFTP.
<i>port</i>	Port number.

4.7.7.3 Example

The following example updates the port of the embedded FTP server.

```
updatePorts('mft_server1', 'FTP', 7021)
```

4.8 MFT Callout Commands

Use the MFT WLST Callout commands, listed in [Table 4-8](#), to manage custom callouts.

Table 4-8 MFT Callout WLST Commands

Use this command...	To...	Use with WLST
createCallouts or <code>crtCalls</code>	Create callouts based on an XML file that defines them.	Online or Offline
deleteCallout or <code>delCalls</code>	Delete a callout.	Online or Offline
listCallouts or <code>lsCalls</code>	List callouts.	Online or Offline
updateCallouts or <code>updCalls</code>	Update callouts with the same names based on an XML file that defines them.	Online or Offline

4.8.1 createCallouts

Command Category: MFT Callout Commands

Use with WLST: Online or Offline

4.8.1.1 Description

Creates callouts based on an XML file that defines them. In the event of an unsupported operation, the command returns a `WLSTException`.

See Processing Transfers with Custom Callouts in *Using Oracle Managed File Transfer* for callout instructions and examples.

4.8.1.2 Syntax

```
createCallouts(def_file_path)
```

```
crtCalls(def_file_path)
```

Argument	Definition
<code>def_file_path</code>	Full path to the callout definition XML file.

4.8.1.3 Example

The following example creates callouts based on a definition file named `/tmp/CalloutDefn.xml`.

```
createCallouts('/tmp/CalloutDefn.xml')
```

4.8.2 deleteCallout

Command Category: MFT Callout Commands

Use with WLST: Online or Offline

4.8.2.1 Description

Deletes a callout. In the event of an unsupported operation, the command returns a `WLSTException`.

4.8.2.2 Syntax

```
deleteCallout(callout_name)
```

```
delCalls(callout_name)
```

Argument	Definition
<i>callout_name</i>	Callout name.

4.8.2.3 Example

The following example deletes a callout named `NotifyTransferComplete`.

```
deleteCallout('NotifyTransferComplete')
```

4.8.3 listCallouts

Command Category: MFT Callout Commands

Use with WLST: Online or Offline

4.8.3.1 Description

Lists callouts. In the event of an unsupported operation, the command returns a `WLSTException`.

4.8.3.2 Syntax

```
listCallouts()
```

```
lsCalls()
```

4.8.3.3 Example

The following example lists callouts.

```
listCallouts()
```

4.8.4 updateCallouts

Command Category: MFT Callout Commands

Use with WLST: Online or Offline

4.8.4.1 Description

Updates callouts with the same names based on an XML file that defines them. In the event of an unsupported operation, the command returns a `WLSTException`.



Note:

Parameters of existing callouts cannot be added, deleted, or modified.

4.8.4.2 Syntax

```
updateCallouts(def_file_path)
```

```
updCalls(def_file_path)
```

Argument	Definition
<i>def_file_path</i>	Full path to the callout definition XML file.

4.8.4.3 Example

The following example updates callouts based on a definition file named `/tmp/CalloutDefn.xml`.

```
updateCallouts('/tmp/CalloutDefn.xml')
```

4.9 MFT Event Notification Commands

Use the MFT WLST Event Notification commands, listed in [Table 4-9](#), to manage contact notifications of events.

Table 4-9 MFT Contact WLST Commands

Use this command...	To...	Use with WLST
addContactToNotification or <code>addContNote</code>	Add a contact to a specific event notification.	Online or Offline
createContact or <code>CrtCont</code>	Create a contact for event notifications.	Online or Offline
deleteContact or <code>delCont</code>	Delete a contact.	Online or Offline
listContacts or <code>lsConts</code>	List contacts.	Online or Offline
removeContactFromNotification or <code>remContNote</code>	Remove a contact from a specific event notification.	Online or Offline
updateEvent or <code>updEvt</code>	Enable or disable a specific event notification.	Online or Offline

4.9.1 addContactToNotification

Command Category: MFT Contact Commands

Use with WLST: Online or Offline

4.9.1.1 Description

Adds a contact to a specific event notification. In the event of an unsupported operation, the command returns a `WLSTException`.

4.9.1.2 Syntax

```
addContactToNotification(event, contact_type, value)
```

```
addContNote(event, contact_type, value)
```

Argument	Definition
<i>event</i>	Event: <code>RUNTIME_ERROR_EVENT</code> , <code>DELETE_ARTIFACT_EVENT</code> , <code>DEPLOY_ARTIFACT_EVENT</code> , <code>EXPORT_IMPORT_EVENT</code> , <code>PURGE_EVENT</code> , or <code>ARCHIVE_RESTORE_EVENT</code> .
<i>contact_type</i>	Contact type: <code>EMAIL</code> , <code>PHONE</code> , <code>FAX</code> , or <code>SMS</code> .
<i>value</i>	Email address or phone number.

4.9.1.3 Example

The following example adds `EMAIL` contact `jane.doe@example.com` to `RUNTIME_ERROR_EVENT` notifications.

```
addContactToNotification('RUNTIME_ERROR_EVENT', 'EMAIL', 'jane.doe@example.com')
```

4.9.2 createContact

Command Category: MFT Contact Commands

Use with WLST: Online or Offline

4.9.2.1 Description

Creates a contact for event notifications. In the event of an unsupported operation, the command returns a `WLSTException`.

See *Configuring an Email Driver for Notifications* in *Using Oracle Managed File Transfer* for information on how to configure an email driver. See *Configuring an SMS Driver for Notifications* in *Using Oracle Managed File Transfer* for information on how to configure an SMS driver.

**Note:**

Phone and FAX notifications are not supported in this release of Oracle Managed File Transfer.

4.9.2.2 Syntax

```
createContact(contact_type, value)
```

```
crtCont(contact_type, value)
```

Argument	Definition
<i>contact_type</i>	Contact type: EMAIL, PHONE, FAX, or SMS.
<i>value</i>	Email address or phone number.

4.9.2.3 Example

The following example creates a contact based on the email address jane.doe@example.com.

```
createContact('EMAIL', 'jane.doe@example.com')
```

4.9.3 deleteContact

Command Category: MFT Contact Commands

Use with WLST: Online or Offline

4.9.3.1 Description

Deletes a contact. In the event of an unsupported operation, the command returns a `WLSTException`.

4.9.3.2 Syntax

```
deleteContact(contact_type, value)
```

```
delCont(contact_type, value)
```

Argument	Definition
<i>contact_type</i>	Contact type: EMAIL, PHONE, FAX, or SMS.
<i>value</i>	Email address or phone number.

4.9.3.3 Example

The following example deletes a contact based on the email address `jane.doe@example.com`.

```
deleteContact('EMAIL', 'jane.doe@example.com')
```

4.9.4 listContacts

Command Category: MFT Contact Commands

Use with WLST: Online or Offline

4.9.4.1 Description

Lists contacts. In the event of an unsupported operation, the command returns a `WLSTException`.

4.9.4.2 Syntax

```
listContacts(contact_type)
```

```
lsConts(contact_type)
```

Argument	Definition
<i>contact_type</i>	Contact type: EMAIL, PHONE, FAX, or SMS.

4.9.4.3 Example

The following example lists email contacts.

```
listContacts('EMAIL')
```

4.9.5 removeContactFromNotification

Command Category: MFT Contact Commands

Use with WLST: Online or Offline

4.9.5.1 Description

Removes a contact from a specific event notification. In the event of an unsupported operation, the command returns a `WLSTException`.

When you remove the last contact from an event, the event is disabled. However, the reverse is not true. When you add the first contact to an event, it is not enabled automatically. Enable an event using the [updateEvent](#) command.

4.9.5.2 Syntax

```
removeContactFromNotification(event, contact_type, value)
```

```
remContNote(event, contact_type, value)
```

Argument	Definition
<i>event</i>	Event: <code>RUNTIME_ERROR_EVENT</code> , <code>DELETE_ARTIFACT_EVENT</code> , <code>DEPLOY_ARTIFACT_EVENT</code> , or <code>EXPORT_IMPORT_EVENT</code> .
<i>contact_type</i>	Contact type: <code>EMAIL</code> , <code>PHONE</code> , <code>FAX</code> , or <code>SMS</code> .
<i>value</i>	Email address or phone number.

4.9.5.3 Example

The following example removes `EMAIL` contact `jane.doe@example.com` from `RUNTIME_ERROR_EVENT` notifications.

```
removeContactFromNotification('RUNTIME_ERROR_EVENT', 'EMAIL', 'jane.doe@example.com')
```

4.9.6 updateEvent

Command Category: MFT Contact Commands

Use with WLST: Online or Offline

4.9.6.1 Description

Enables or disables a specific event notification. Event notifications are disabled by default. In the event of an unsupported operation, the command returns a `WLSTException`.

4.9.6.2 Syntax

```
updateEvent(event, enabled)
```

```
updEvt(event, enabled)
```

Argument	Definition
<i>event</i>	Event: <code>RUNTIME_ERROR_EVENT</code> , <code>DELETE_ARTIFACT_EVENT</code> , <code>DEPLOY_ARTIFACT_EVENT</code> , or <code>EXPORT_IMPORT_EVENT</code> .
<i>enabled</i>	Enabled: <code>TRUE</code> or <code>FALSE</code> .

4.9.6.3 Example

The following example disables `DELETE_ARTIFACT_EVENT` notifications.

```
updateEvent('DELETE_ARTIFACT_EVENT', 'FALSE')
```

4.10 MFT Archive and Restore Commands

Use the MFT Archive and Restore commands, listed in [Table 4-10](#), to archive and restore runtime instances or file system data based on criteria such as status and date ranges.

Table 4-10 MFT Archive and Restore WLST Commands

Use this command...	To...	Use with WLST
archiveInstanceData or <code>arcData</code>	Archive runtime instances based on specified criteria.	Online or Offline
restoreInstanceData or <code>resData</code>	Restore previously archived runtime instances.	Online or Offline
archivePayloads or <code>arcPLs</code>	Archive file system data corresponding to runtime instance data based on specified criteria.	Online or Offline
restorePayloadsByName or <code>resPLbyN</code>	Restore previously archived file system data by the zip file name.	Online or Offline
restorePayloadsByPrefix or <code>resPLbyP</code>	Restore previously archived file system data by the zip file name prefix.	Online or Offline

4.10.1 archiveInstanceData

Command Category: MFT Archive and Restore Commands

Use with WLST: Online or Offline

4.10.1.1 Description

Archives runtime instances based on specified criteria. Optionally archives file system data. In the event of an unsupported operation, the command returns a `WLSTException`.

Before you can run this command, you must create an MFT archive directory on the database server using the following SQL commands:

```
CREATE DIRECTORY MFT_DIR AS path;  
GRANT READ, WRITE ON DIRECTORY MFT_DIR TO MFT_STB;
```

For more information about the database and the schema owner (MFT_STB), see *Installing and Configuring Managed File Transfer*.

4.10.1.2 Syntax

```
archiveInstanceData(archiveFileName, startDate, endDate, testMode, batchId, status,  
comments, runInSync, transferNames, namesDelimiter)
```

```
arcData(archiveFileName, startDate, endDate, testMode, batchId, status, comments,  
runInSync, transferNames, namesDelimiter)
```

Argument	Definition
<i>archiveFileName</i>	Name of the runtime instance archive file to be saved to the defined MFT archive directory. The default extension is <code>.dmp</code> . This argument is required.
<i>startDate</i>	Timestamp in format <code>dd-MM-yyyy H:m:s:S</code> . Default is no date and time specified. Expressions are not supported.
<i>endDate</i>	Timestamp in format <code>dd-MM-yyyy H:m:s:S</code> . Default is the current date and time. Expressions are not supported.
<i>batchId</i>	Identifier in the output of a previous <code>archiveInstanceData</code> command. Default input is an empty string. Use to rerun a tested or failed <code>archiveInstanceData</code> from the point of failure or to run archivePayloads .
<i>status</i>	Status: <code>*</code> for all statuses, <code>C</code> for COMPLETED, <code>F</code> for FAILED, or <code>A</code> for ACTIVE (in progress). Default is <code>C</code> . You can specify multiple comma-separated statuses.
<i>testMode</i>	If passed <code>true</code> , no instance data is archived. It can be used just to get info like, how many records will be archived for given criteria.
<i>comments</i>	Comment string. Default is an empty string.
<i>runInSync</i>	If actual archive is to be executed as background process or in current thread itself. If passed <code>false</code> , command will return after capturing the snapshot & actual archive will be executed as a background process.
<i>transferNames</i>	Delimiter separated string containing names of transfers whose instances need to be archived.
<i>namesDelimiter</i>	Character used as separator between names of artifacts.

4.10.1.3 Example

The following example runs in test mode (does not archive anything) and in the background (does not block WLST commands while running) for all instances with status `COMPLETED`, and provides statistics such as how many instances or payloads would be archived.

```
archiveInstanceData()
```

Here is an example of the output:

```
Total no. of instances to be archived: 105.
Total no. of payloads to be archived: 105.
Nothing has been archived as test mode was TRUE.
To archive these records, run the same command
by passing batch id 546781 and test mode as FALSE.
```

The following example runs in test mode and in the background for instances before March 31st, 2013 with status `COMPLETED`, and provides archive statistics. Note that when the *startDate* argument is omitted, the *endDate* argument must be explicitly labeled.

```
archiveInstanceData(endDate='31-03-2013 00:00:00:00')
```

The following example archives instances from February 1st, 2013 to March 31st, 2013 with status `COMPLETED` to the file `mft.dmp`, and provides archive statistics. Note that when the `startDate` and `endDate` arguments are both included, labels may be omitted.

```
archiveInstanceData(archiveFileName='mft.dmp', '01-02-2013 00:00:00:00', '31-03-2013
00:00:00:00', testMode='FALSE')
```

The following example archives instances from February 1st, 2013 to March 31st, 2013 with status `ACTIVE` or `FAILED` to the file `mft.dmp`, and provides archive statistics. Because `runInSync='TRUE'` is specified, the archive occurs immediately and blocks further `WLST` commands until completion. Because `runFSArchive='TRUE'` is specified, file system data is also archived.

```
archiveInstanceData(archiveFileName='mft.dmp', '01-02-2013 00:00:00:00', '31-03-2013
00:00:00:00', testMode='FALSE', status='A,F',
runInSync='TRUE', fsArchiveFolderPath='2013-01-02-MftArchive')
```

The following example runs a previously tested or failed archive with batch ID 546781 from the point of failure.

```
archiveInstanceData(archiveFileName='mft2.dmp', batchId='546781', testMode='FALSE')
```

4.10.2 restoreInstanceData

Command Category: MFT Archive and Restore Commands

Use with `WLST`: Online or Offline

4.10.2.1 Description

Restores previously archived runtime instances. In the event of an unsupported operation, the command returns a `WLSTException`.

4.10.2.2 Syntax

```
restoreInstanceData(archiveFileName, fileNamePrefix, fsFolderPath)
```

```
resData(archiveFileName, fileNamePrefix, fsFolderPath)
```

Argument	Definition
<i>archiveFilePath</i>	Name of the archive file in the MFT archive directory to be restored. For details about how this directory is defined, see archiveInstanceData . The default extension is <code>.dmp</code> . This argument is required.
<i>fileNamePrefix</i>	String to identify the file name prefix of the corresponding payload archive files, usually the batch ID. You must have run archivePayloads with the archiveInstanceData batch ID first. No default.
<i>fsFolderPath</i>	String to identify the path to the directory where the corresponding payload archive files are stored. No default.

4.10.2.3 Example

The following example restores runtime instance data previously archived by an [archiveInstanceData](#) command. The corresponding payload archive is restored from `/tmp/mft/546781.zip`. This command runs in the background.

```
restoreInstanceData(archiveFilePath="mft_runtime_archive.dmp",
fileNamePrefix="546781", fsFolderPath="/tmp/mft")
```

4.10.3 archivePayloads

Command Category: MFT Archive and Restore Commands

Use with WLST: Online or Offline

4.10.3.1 Description

Archives file system data corresponding to runtime instance data based on specified criteria. In the event of an unsupported operation, the command returns a `WLSTException`.

4.10.3.2 Syntax

```
archivePayloads(arguments)
```

```
arcPLs(arguments)
```

Argument	Definition
<i>batchId</i>	String to identify the archive criteria. No default: obtain a <i>batchId</i> by running archiveInstanceData first. This argument is required.
<i>archivePath</i>	Path to the directory to which the <i>batchId.zip</i> archive is saved. This directory must exist. A very large archive may be saved in multiple files named <i>batchId_n.zip</i> . This argument is required.
<i>runInSync</i>	Whether to run immediately, synchronously, and return (TRUE), or run in the background, asynchronously, allowing execution of other WLST commands (FALSE). Default is FALSE.

4.10.3.3 MBean Properties

For details about how to set MBean properties using Enterprise Manager, see [Setting System MBean Properties for MFT WLST Commands](#).

Property	Definition
<i>FS_ARCHIVE_MAX_SIZE</i>	String to set a size limit in MB for archive zip files. The default value is zero, which sets no limit.
<i>FS_ARCHIVE_MAX_FILES_PER_ZIP</i>	String to set a limit on the number of files an archive zip file can contain. The default value is zero, which sets no limit.

4.10.3.4 Example

The following example archives file system data corresponding to the runtime instance data previously archived by an [archiveInstanceData](#) command with the batch ID 546781. The archive is saved to `/tmp/mft/546781.zip` and runs immediately.

```
archivePayloads(batchId='546781',archivePath='/tmp/mft',runInSync='TRUE')
```

4.10.4 restorePayloadsByName

Command Category: MFT Archive and Restore Commands

Use with WLST: Online or Offline

4.10.4.1 Description

Restores previously archived file system data. In the event of an unsupported operation, the command returns a `WLSTException`.

4.10.4.2 Syntax

```
restorePayloadsByName(arguments)
```

```
resPLbyN(arguments)
```

Argument	Definition
<i>fileNames</i>	String to identify the names of the <code>.zip</code> files to restore. No default. This argument is required. If the <code>.zip</code> file names have been changed to something other than the batch IDs, you must specify the new file names instead of the batch IDs for this argument. For an archive that consists of multiple <code>batchId_n.zip</code> or <code>prefix_n.zip</code> files, you can use the restorePayloadsByPrefix command instead.
<i>folderPath</i>	Path to the directory in which the <code>.zip</code> archives were saved. This argument is required.
<i>runInSync</i>	Whether to run immediately, synchronously, and return (<code>TRUE</code>), or run in the background, asynchronously, allowing execution of other WLST commands (<code>FALSE</code>). Default is <code>FALSE</code> .

4.10.4.3 MBean Properties

For details about how to set MBean properties using Enterprise Manager, see [Setting System MBean Properties for MFT WLST Commands](#).

Property	Definition
<code>FS_RESTORE_OVERWRITE_EXISTING</code>	String to determine whether a restore overwrites files having the same name. The default value is <code>TRUE</code> .

4.10.4.4 Example

The following example restores file system data corresponding to the runtime instance data previously archived by an [archivePayloads](#) command with the batch ID 546781. The archive is restored from `/tmp/mft/546781.zip` and runs immediately.

```
restorePayloadsByName(fileName='546781',folderPath='/tmp/mft',runInSync='TRUE')
```

4.10.5 restorePayloadsByPrefix

Command Category: MFT Archive and Restore Commands

Use with WLST: Online or Offline

4.10.5.1 Description

Restores previously archived file system data. In the event of an unsupported operation, the command returns a `WLSTException`.

4.10.5.2 Syntax

```
restorePayloadsByPrefix(arguments)
```

```
resPLbyP(arguments)
```

Argument	Definition
<i>fileNamePrefix</i>	String to identify the <code>.zip</code> files to restore, usually the batch ID. No default: obtain a <code>batchId</code> by running archivePayloads first. This argument is required. If the <code>.zip</code> file names have been changed to something other than the batch ID, you must specify the new file name instead of the batch ID for this argument. If you rename one <code>prefix_n.zip</code> file you must rename all, without changing the <code>_n</code> portion, or this command will not restore the entire archive.
<i>folderPath</i>	Path to the directory in which the <code>.zip</code> archives were saved. This argument is required.
<i>runInSync</i>	Whether to run immediately, synchronously, and return (<code>TRUE</code>), or run in the background, asynchronously, allowing execution of other WLST commands (<code>FALSE</code>). Default is <code>FALSE</code> .

4.10.5.3 MBean Properties

For details about how to set MBean properties using Enterprise Manager, see [Setting System MBean Properties for MFT WLST Commands](#).

Property	Definition
<code>FS_RESTORE_OVERWRITE_EXISTING</code>	String to determine whether a restore overwrites files having the same name. The default value is <code>TRUE</code> .

4.10.5.4 Example

The following example restores file system data corresponding to the runtime instance data previously archived by an `archivePayloads` command with the batch ID 546781. The archive is restored from `/tmp/mft/546781.zip` and runs immediately.

```
restorePayloadsByPrefix(fileNamePrefix='546781',folderPath='/tmp/
mft',runInSync='TRUE')
```

4.11 MFT Purge Commands

Use the MFT Purge commands, listed in [Table 4-11](#), to purge runtime instances or file system data based on criteria such as status and date ranges.

Table 4-11 MFT Purge WLST Commands

Use this command...	To...	Use with WLST
<code>purgeInstanceData</code> or <code>prgData</code>	Purge runtime instances based on specified criteria. Optionally purges file system data.	Online or Offline
<code>purgePayloads</code> or <code>prgPLs</code>	Purge file system data corresponding to runtime instance data based on specified criteria.	Online or Offline

4.11.1 `purgeInstanceData`

Command Category: MFT Purge Commands

Use with WLST: Online or Offline

4.11.1.1 Description

Purges runtime instances based on specified criteria. Optionally purges file system data. In the event of an unsupported operation, the command returns a `WLSTException`.

4.11.1.2 Syntax

```
purgeInstanceData(startDate, endDate, testMode, batchId, status, comments,
runInSync, runPayloadPurge, transferNames, namesDelimiter)
```

```
prgData(startDate, endDate, testMode, batchId, status, comments, runInSync,
runPayloadPurge, transferNames, namesDelimiter)
```

Argument	Definition
<code>startDate</code>	Timestamp in format <code>dd-MM-yyyy H:m:s:S</code> . Default is no date and time specified. Expressions are not supported.
<code>endDate</code>	Timestamp in format <code>dd-MM-yyyy H:m:s:S</code> . Default is the current date and time. Expressions are not supported.

Argument	Definition
<i>batchId</i>	Identifier in the output of a previous <code>purgeInstanceData</code> command. Default input is an empty string. Use to rerun a tested or failed <code>purgeInstanceData</code> from the point of failure or to run purgePayloads .
<i>status</i>	Status: * for all statuses, C for COMPLETED, F for FAILED, or A for ACTIVE (in progress). Default is C. You can specify multiple comma-separated statuses.
<i>testMode</i>	Whether what to purge is tested but not purged: TRUE or FALSE. Default is TRUE. Therefore, to actually purge runtime instances, you must explicitly set <code>testMode=FALSE</code> .
<i>comments</i>	Comment string. Default is an empty string.
<i>runInSync</i>	Whether to run immediately, synchronously, and return (TRUE), or run in the background, asynchronously, allowing execution of other WLST commands (FALSE). Default is FALSE.
<i>runPayloadPurge</i>	Whether to also purge file system data corresponding to runtime instance data: TRUE or FALSE. Default is FALSE. To purge file system data separately, use purgePayloads .
<i>transferNames</i>	Delimiter separated string containing names of transfers whose instances need to be purged.
<i>namesDelimiter</i>	Character used as separator between names of artifacts.

4.11.1.3 Example

The following example runs in test mode (does not purge anything) and in the background (does not block WLST commands while running) for all instances with status `COMPLETED`, and provides statistics such as how many instances or payloads would be purged.

```
purgeInstanceData()
```

Here is an example of the output:

```
Total no. of purgeable instances: 105.
Total no. of purgeable payloads: 105.
Nothing has been purged as test_mode was TRUE.
To purge these records, run the same command
by passing batchId 546781 and test mode as FALSE.
```

The following example runs in test mode and in the background for instances before March 31st, 2013 with status `COMPLETED`, and provides purge statistics. Note that when the `startDate` argument is omitted, the `endDate` argument must be explicitly labeled.

```
purgeInstanceData(endDate='31-03-2013 00:00:00:00')
```

The following example purges instances from February 1st, 2013 to March 31st, 2013 with status `COMPLETED`, and provides purge statistics. Note that when the `startDate` and `endDate` arguments are both included, labels may be omitted.

```
purgeInstanceData('01-02-2013 00:00:00:00', '31-03-2013 00:00:00:00',
testMode='FALSE')
```

The following example purges instances from February 1st, 2013 to March 31st, 2013 with status `ACTIVE` or `FAILED`, and provides purge statistics. Because `runInSync='TRUE'` is specified, the purge occurs immediately and blocks further WLST commands until completion. Because `runPayloadPurge='TRUE'` is specified, file system data is also purged.

```
purgeInstanceData('01-02-2013 00:00:00:00', '31-03-2013 00:00:00:00',
testMode='FALSE', status='A,F', runInSync='TRUE',
runPayloadPurge='TRUE')
```

The following example runs a previously tested or failed purge with batch ID 546781 from the point of failure.

```
purgeInstanceData(batchId='546781', testMode='FALSE')
```

4.11.2 purgePayloads

Command Category: MFT Purge Commands

Use with WLST: Online or Offline

4.11.2.1 Description

Purges file system data corresponding to runtime instance data based on specified criteria. In the event of an unsupported operation, the command returns a `WLSTException`.

4.11.2.2 Syntax

```
purgePayloads(batchId, detailedAudit, runInSync)
```

```
prgPLs(batchId, detailedAudit, runInSync)
```

Argument	Definition
<i>batchId</i>	String to identify the purge criteria. No default: obtain a <code>batchId</code> by running <code>purgeInstanceData</code> first.
<i>detailedAudit</i>	Whether each purged file is audited: <code>TRUE</code> or <code>FALSE</code> . Default is <code>TRUE</code> .
<i>runInSync</i>	Whether to run immediately, synchronously, and return (<code>TRUE</code>), or run in the background, asynchronously, allowing execution of other WLST commands (<code>FALSE</code>). Default is <code>FALSE</code> .

4.11.2.3 MBean Properties

For details about how to set MBean properties using Enterprise Manager, see [Setting System MBean Properties for MFT WLST Commands](#).

Property	Definition
<i>FS_PURGE_BATCH_SIZE</i>	String to determine the batch size for reading payload paths. The <code>purgePayloads</code> command reads the payload paths from the database in batches to avoid running out of memory. The default value is 5000.
<i>FS_PURGE_TEMP_TABLES_LIFE</i>	String to determine the minimum age in days of temporary tables that are deleted after a payload purge. The default value is 30.
<i>FS_PURGE_THREAD_COUNT</i>	String to determine how many threads per node are used for performing the payload purge. The default value is 1.

4.11.2.4 Example

The following example purges file system data corresponding to the runtime instance data previously purged by a `purgeInstanceData` command with the batch ID 546781. Detailed audits of purged files are performed, and the purge runs in the background.

```
purgePayloads(batchId='546781')
```

The following example purges file system data corresponding to the runtime instance data previously purged by a `purgeInstanceData` command with the batch ID 546781. Detailed audits of purged files are **not** performed, and the purge runs in the background.

```
purgePayloads(batchId='546781', detailedAudit='FALSE')
```

The following example purges file system data corresponding to the runtime instance data previously purged by a `purgeInstanceData` command with the batch ID 546781. Detailed audits of purged files are performed, and the purge occurs immediately.

```
purgePayloads(batchId='546781', detailedAudit='TRUE', runInSync='TRUE')
```

4.12 Setting System MBean Properties for MFT WLST Commands

Some MFT WLST commands have associated properties you can set to refine the operations they perform.

The steps for this process are:

1. Log in to the Oracle Enterprise Manager console.
2. In the Target Navigation pane, expand the Weblogic Domain node.
3. Select the domain on which the Oracle WebLogic Server managed server dedicated to Oracle Managed File Transfer is installed.

For example, the domain might be `soainfra` or `base_domain`.

4. Right-click on the domain and select **System MBean Browser**.
5. Click the Find icon.
6. Type `MFTConfig` in the **MBean Name** text box. Press Enter.

The Application Defined MBeans: MFTConfig: mft page appears.

7. Click the **Operations** tab.
8. Click **addProperty**.

The Operation: addProperty page appears.

9. Type the property name in the **key** text box. Type the property value in the **value** text box. Typing a **comment** is optional. Click **Invoke**.
10. To verify that the property has been added:
 - a. Click **Return**.
 - b. Click the **Attributes** tab.
 - c. Click **Properties**.

The Attribute: Properties page appears. The property you added should be listed.