

Oracle® Fusion Middleware

Using Oracle WebLogic Server Multitenant (Deprecated)



12c (12.2.1.4.0)

E90793-01

September 2019

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

ORACLE®

Oracle Fusion Middleware Using Oracle WebLogic Server Multitenant (Deprecated), 12c (12.2.1.4.0)

E90793-01

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	xvi
Documentation Accessibility	xvi
Related Documents	xvi
Conventions	xvii
New and Changed Features in This Release	xvii

1 Oracle WebLogic Server Multitenant

About WebLogic Server MT	1-2
WebLogic Server MT Supports Only Java EE Applications	1-3
Key Concepts in WebLogic Server MT	1-3
Tenants	1-3
Resource Groups	1-4
Resource Group Templates	1-4
Virtual Targets	1-5
Scopes	1-5
Resource Isolation	1-6
WebLogic Server MT Administrators	1-7
Understanding SaaS Multitenancy	1-7
Testing Applications in a SaaS Environment	1-9
Understanding PaaS Multitenancy	1-9
Lifecycle Management	1-10

2 Configuring Oracle WebLogic Server Multitenant

How to Manage WebLogic Server MT	2-1
Configuring WebLogic Server MT	2-1

3 Configuring Virtual Targets

About Virtual Targets	3-1
Components of a Virtual Target	3-2

Virtual Targets and Load Balancing	3-3
Using Partition Channels with Virtual Targets	3-3
About Partition Administrative Virtual Targets	3-4
Creating Virtual Targets: Main Steps and Examples	3-5
Creating Virtual Targets: WLST Example	3-6
Creating Virtual Targets: REST Example	3-6
Managing Virtual Targets: Main Steps and Examples	3-6
Deleting Virtual Targets in Use by a Partition	3-7
Deleting Virtual Targets in Use by a Resource Group at the Domain Level	3-8
Managing Virtual Targets: WLST Example	3-8
Managing Virtual Targets: REST Example	3-9
Configuring Virtual Targets: Related Tasks and Links	3-10

4 Configuring Resource Group Templates

About Resource Group Templates	4-1
What Is the Difference Between a Resource Group Template and a Resource Group?	4-2
What Is in a Resource Group Template?	4-2
Resource Group Templates and Overrides	4-2
Resource Group Template Example	4-3
Creating Resource Group Templates: Main Steps and WLST Examples	4-4
Creating Resource Groups: WLST Example	4-4
Configuring Resource Group Templates: Main Steps and WLST Examples	4-5
Configuring Resource Group Template Deployment Settings	4-5
Configuring Resource Group Template Services Settings	4-5
Configuring Resource Group Template JDBC Settings	4-6
Configuring Resource Group Template JMS settings	4-6
Configuring Resource Group Mail Session Settings	4-10
Configuring Resource Group Persistent Store Settings	4-10
Configuring Resource Group Foreign JNDI Provider Settings	4-10
Configuring Resource Group Diagnostic System Module Settings	4-11
Configuring Resource Group Template Notes	4-11
Configuring Resource Group Template: WLST Example	4-12
Deleting Resource Group Templates: Main Steps and WLST Examples	4-12
Deleting Resource Group Templates: WLST Example	4-13
Configuring Resource Group Templates: Related Tasks and Links	4-14

5 Configuring Security

New Security Features in Domain Partitions	5-1
Domain Partition Security Realms: Overview	5-2

Identity Domains: Overview	5-3
Types of Identity Domains	5-3
Default Identity Domain Values	5-4
Administrative Roles for Configuration and Management	5-4
Configuring the Administrative Identity Domain: Main Steps and WLST Example	5-5
Configuring the Administrative Identity Domain: WLST Example	5-6
Configuring Security Realms and Primary Identity Domains: Main Steps and Examples	5-6
Configuring Security Realms and Primary Identity Domains: WLST Example	5-8
Configuring Security Realms and Primary Identity Domains: REST Example	5-10
Setting the Identity Domain-Aware Providers Required Control: Main Steps	5-10
Connecting Directly to Domain Partitions: Main Steps and WLST Example	5-11
Connecting Directly to Domain Partitions: WLST Example	5-11
Managing Security Data as a Partition Administrator: Main Steps and Examples	5-13
Managing Users and Groups as a Partition Administrator: Fusion Middleware Control Example	5-14
Creating a New User as a Partition Administrator: WLST Example	5-15
Managing Users and Groups as a Partition Administrator: REST Example	5-17
Configuring SSL in Domain Partitions	5-17
Configuring Security in Domain Partitions: Related Tasks and Links	5-17

6 Configuring Oracle Traffic Director

Configuring Oracle Traffic Director: Overview	6-1
Oracle Traffic Director Partitions	6-2
Monitoring	6-2
Logging	6-2
Configuring Oracle Traffic Director: Main Steps	6-3
Creating the Domain for Oracle Traffic Director	6-3
Creating an Oracle Traffic Director MT Configuration and Instance	6-4
Using Fusion Middleware Control to Create the Configuration and Instance	6-4
Using WLST to Create the Configuration and Instance	6-4
Registering the Oracle Traffic Director Runtime	6-4
Oracle Traffic Director: Troubleshooting	6-5
Frequently Asked Questions	6-5
Configuring Oracle Traffic Director: Related Information	6-7

7 Configuring Domain Partitions

Creating Domain Partitions	7-1
Creating Domain Partitions: Prerequisites	7-1
Oracle Traffic Director: WebLogic Server Plug-in Enabled Prerequisite	7-3

Creating Domain Partitions: Main Steps and Examples	7-4
Creating Domain Partitions: WLST Example	7-5
Creating Domain Partitions: REST Example	7-6
Managing Domain Partitions: Main Steps and Examples	7-6
Managing Domain Partitions: WLST Example	7-6
Managing Domain Partitions: REST Example	7-7
Controlling Domain Partitions: Main Steps and Examples	7-7
Actions That Require a Partition Restart	7-8
Controlling Domain Partitions: WLST Example	7-8
Controlling Domain Partitions: REST Example	7-9
Configuring Domain Partitions: Related Tasks and Links	7-10

8 Configuring Resource Groups

Configuring Resource Groups: Overview	8-1
What Is in a Resource Group?	8-2
Resource Groups and Overrides	8-2
Resource Groups in Global Scope and Partitions	8-2
Targeting a Resource Group to More Than One Target	8-3
Creating Resource Groups: Main Steps and Examples	8-3
Creating Resource Groups: WLST Example	8-4
Creating Resource Groups: REST Example	8-5
Configuring Resource Groups: Main Steps and Examples	8-5
Configuring Resource Group General Settings	8-5
Configuring Resource Group Deployment Settings	8-5
Deploying Applications to a Resource Group	8-6
Redeploying Applications to a Resource Group	8-6
Undeploying Applications from a Resource Group	8-7
Configuring Resource Group Services Settings	8-7
Configuring Resource Group JDBC Settings	8-7
Configuring Resource Group JMS settings	8-8
Configuring Resource Group Mail Session Settings	8-11
Configuring Resource Group Persistent Store Settings	8-12
Configuring Resource Group Foreign JNDI Provider Settings	8-12
Configuring Resource Group Diagnostic System Module Settings	8-13
Configuring Resource Group Targets	8-13
Configuring Resource Group Notes	8-13
Configuring Resource Groups: WLST Example	8-14
Configuring Resource Groups: REST Example	8-15
Deleting Resource Groups: Main Steps and WLST Example	8-15
Deleting Resource Groups: WLST Example	8-15

Controlling Resource Groups: Main Steps and WLST Example	8-16
Controlling Resource Groups: WLST Example	8-16
Migrating Resource Groups: Main Steps and WLST Example	8-17
Migrating Resource Groups: WLST Example	8-17
Managing Administrative Applications and Resources in Partition Resource Groups	8-18
Creating Administrative Resource Groups	8-19
Targeting Administrative Resource Groups	8-20
About Partition Lifecycle States and Transitions	8-20
Configuring Resource Groups: Related Tasks and Links	8-22

9 Configuring Resource Overrides

Resource Overrides: Overview	9-1
Using Configuration MBean Overrides	9-2
Using Resource Deployment Plans	9-3
resource-deployment-plan Syntax	9-3
Sample Resource Deployment Plan	9-5
Configuring Resource MBean Overrides: Main Steps and WLST Examples	9-6
Configuring a JDBC System Resource Override: Main Steps	9-6
Configuring a JDBC System Resource Override: WLST Example	9-6
Configuring a JMS System Resource Override: Main Steps	9-6
Configuring a JMS System Resource Override: WLST Example	9-7
Configuring a Mail Session Override: Main Steps	9-8
Configuring a Mail Session Override: WLST Example	9-8
Configuring Resource Deployment Plans: Main Steps and WLST Example	9-8
Configuring Resource Overrides: Related Tasks and Links	9-9

10 Configuring Resource Consumption Management

Configuring Resource Consumption Management: Overview	10-1
Software Requirements for Using RCM	10-2
Why Do You Need Resource Consumption Management?	10-2
How to Enable RCM	10-2
Supported Resources for RCM	10-2
Configuring Resource Consumption Management: Main Steps	10-3
Triggers	10-3
Fair Share	10-5
Determining Fair Share Allocations for a Resource	10-6
Creating a Resource Manager	10-6
Example RCM Configuration in config.xml	10-6
Dynamic Reconfiguration of Resource Managers	10-8

Configuring Resource Consumption Management: Monitoring Resource Use	10-8
Best Practices and Considerations When Using Resource Consumption Management	10-9
General Considerations	10-9
Monitor Average and Peak Resource Use	10-9
When to Use a Trigger	10-9
When to Use Fair Share	10-10
Use Complementary Workloads	10-10
When to Use Partition-Scoped RCM Policies	10-10
Managing CPU Use	10-10
Managing Heap	10-11
RCM Limitations	10-11
Configuring Resource Consumption Management: WLST Example	10-11
RCM WLST Example: Overview	10-11
RCM WLST Example: WLST Script	10-12
Configuring Resource Sharing: Related Tasks and Links	10-13

11 Deploying Applications

About Deployment Scopes	11-1
Deploying Applications to Resource Group Templates	11-2
Deploying Applications to Resource Group Templates: Main Steps	11-2
Deploying Applications to Resource Group Templates: Examples	11-3
Deploying Applications to Resource Group Templates: WLST Example	11-3
Redeploying Applications to Resource Group Templates: WLST Example	11-3
Undeploying Applications from Resource Group Templates: WLST Example	11-3
Updating Applications in Resource Group Templates: WLST Example	11-4
Distributing Applications to Resource Group Templates: WLST Example	11-4
Deploying Applications to Resource Group Templates: Related Tasks and Links	11-4
Deploying Applications to Partition Resource Groups	11-5
Deploying Applications to Partition Resource Groups: Main Steps	11-6
Deploying Applications to Partition Resource Groups: Examples	11-7
Deploying Applications to Partition Resource Groups: WLST Example	11-7
Redeploying Applications to Partition Resource Groups: WLST Example	11-7
Undeploying Applications from Partition Resource Groups: WLST Example	11-7
Updating Applications in Partition Resource Groups: WLST Example	11-8
Distributing Applications to Partition Resource Groups: WLST Example	11-8
Starting Applications on Partition Resource Groups: WLST Example	11-8
Stopping Applications on Partition Resource Groups: WLST Example	11-8
Deploying Applications to Partition Resource Groups: REST Example	11-8
Deploying Applications to Partition Resource Groups: Related Tasks and Links	11-9
Deploying Applications as the Partition Administrator	11-9

Overriding Application Configuration	11-10
Overriding Application Configuration: Main Steps	11-11
Overriding Application Configuration: WLST Example	11-11
Overriding Application Configuration: Related Tasks and Links	11-12
Removing an Application Override	11-13
Removing an Application Override: Main Steps	11-13
Removing an Application Override: WLST Example	11-13
Removing an Application Override: Related Tasks and Links	11-13
Using Partition-Specific Deployment Plans	11-14
Using Partition-Specific Deployment Plans: Main Steps	11-14
Using Partition-Specific Deployment Plans: WLST Example	11-15
Using Partition-Specific Deployment Plans: Related Links and Tasks	11-15
Enabling Parallel Deployment in Multitenant Environments	11-16

12 Configuring the Shared Application Classloader

Configuring the Shared Application Classloader: Overview	12-1
Configuring the Shared Application Classloader: Main Steps	12-2
Configuring the Shared Application Classloader: Related Tasks and Links	12-3

13 Configuring Oracle Coherence

Configuring Oracle Coherence: Overview	13-1
Deploying Oracle Coherence Applications in Multitenant Domains	13-2
Overriding Cache Configuration Properties	13-2
Enabling Cache Sharing Across Partitions	13-3
Securing Oracle Coherence Applications in Multitenant Domains	13-4
Configuring Oracle Coherence: Related Tasks and Links	13-5

14 Configuring JDBC

Configuring JDBC: Prerequisites	14-1
About Supported Data Source Types	14-2
Configuring JDBC Data Sources: Considerations	14-4
Data Source Scope	14-4
Runtime Monitoring	14-4
Security	14-5
Transactions Scope	14-5
Partition Life Cycle	14-5
Diagnostic Image Source	14-6
Logging	14-6
JNDI Bindings	14-6

Deprecated Drivers Not Supported	14-6
Configuring JDBC Data Sources: WLST Example	14-7
Configuring JDBC Data Sources: Administration Console Example	14-9
Configuring JDBC Data Sources	14-9
Monitoring JDBC Data Sources	14-10
Configuring JDBC Data Source Diagnostics	14-10
Configuring Partition-Scoped Deployments	14-11
Using Resource Deployment Plans	14-13
Configuring JDBC Data Sources: Fusion Middleware Control Example	14-14
Configuring JDBC Data Sources: REST Example	14-15
Configuring JDBC Data Sources: Related Tasks and Links	14-17

15 Configuring Messaging

Configuring Messaging: Prerequisites	15-1
About Messaging Configuration Scopes	15-2
About Configuration Validation and Targeting Rules	15-3
Configuring Messaging Components	15-3
Configuring JDBC or File Persistent Stores	15-4
Configuring JMS Servers	15-5
Configuring Store-and-Forward Agents	15-5
Configuring Path Services to Support Using Unit-of-Order with Distributed Destinations	15-6
Configuring Messaging Bridges	15-6
Configuring JMS System Resources and Application-Scoped JMS Modules	15-7
Configuring Partition-Specific JMS Overrides	15-9
Accessing Partition-Scoped Messaging Resources Using JNDI	15-11
Specifying No URL	15-11
Specifying a Partition Virtual Host or Partition URI	15-11
Specifying a Dedicated Port URL	15-12
Local Cross-Partition Use Cases Using local: URLs or Decorated JNDI Names	15-12
About Partition Associations in JMS	15-12
Partition Association Between Connection Factories and Their Connections or JMS Contexts	15-13
Partition Association with Asynchronous Callbacks	15-13
Connection Factories and Destinations Need Matching Scopes	15-13
Temporary Destination Scoping	15-13
Managing Partition-Scoped Messaging Components	15-14
Runtime Monitoring and Control	15-14
Managing Partition-Scoped Security	15-14
Managing Transactions	15-15
Managing Partition and Resource Group Lifecycle Operations	15-15

Partition-Scoped JMS Diagnostic Image Sources	15-15
Partition-Scoped JMS Logging	15-15
Message Lifecycle Logging	15-15
Admin Helpers	15-16
File Locations	15-17
Configuring Messaging: Best Practices	15-19
Configuring Messaging: Limitations	15-19
Messaging Resource Group Migration	15-21
Resource Group Migration with Persistent Data	15-22
Migrating Persistent Data for a Non-Clustered Service	15-22
Migrating Persistent Data for a Cluster Singleton Service	15-22
Migrating Persistent Data for a Cluster Distributed Service	15-23
Migrating Store-and-Forward Messages	15-24
Migrating Message Driven Beans (MDBs)	15-24
Global Transaction Considerations With Third Party JMS	15-25
Client Failover During Resource Group Migration	15-25
Manual Failover	15-26
Using Oracle Traffic Director TCP Proxy	15-26

16 Configuring and Programming JNDI

Configuring Foreign JNDI Providers: Overview	16-1
Configuring Foreign JNDI Providers: Prerequisites	16-1
Configuring Foreign JNDI Providers: Main Steps	16-2
Creating Foreign JNDI Providers: WLST Example	16-2
Programming JNDI in a Partitioned Environment	16-2
Introduction to Partition-Scoped and Domain-Scoped JNDI Resources	16-3
Object-Based Partition Association	16-4
Obtaining the Partition Information of a Context	16-4
Accessing JNDI Resources Remotely and Across Partitions	16-4
Cross-Partition Authentication	16-4
Clustered JNDI in a Partitioned Environment	16-4
Life Cycle of a Partitioned JNDI Resource	16-5
Creating Foreign JNDI Providers: Related Tasks and Links	16-5

17 Configuring Partition Work Managers

Partition Work Managers: Overview	17-1
Configuring Partition Work Managers: Main Steps	17-2
Defining Partition Work Managers: WLST Example	17-3
Configuring Domain-Level Partition Work Managers: WLST Example	17-3

Associating Partition Work Managers with Partitions: WLST Example	17-3
Defining Partition Work Manager Attributes in a Partition: WLST Example	17-4
Configuring Partition Work Managers: Related Tasks and Links	17-5

18 Migrating a WebLogic Server Domain to a Domain Partition

Migrating a WebLogic Server Domain to a Domain Partition: Overview	18-1
New features of the Domain to Partition Conversion Tool	18-2
Migrating a WebLogic Server Domain to a Domain Partition: Prerequisites	18-2
Migrating a WebLogic Server Domain to a Domain Partition: Main Steps	18-3
Preparing to Export the WebLogic Server Domain Applications Environment	18-3
Exporting the WebLogic Server Domain Applications Environment	18-4
Using the JSON File to Override Defaults During Import	18-7
Importing an Applications Archive File to a Domain Partition	18-11
Monitoring the Export and Import Operation using Reports	18-12
Migrating a WebLogic Server Domain to a Domain Partition: Limitations and Considerations	18-13
Considerations and Limitations for Importing a Domain with JMS Resources	18-13

19 Configuring Partition Concurrent Managed Objects

Configuring Partition Concurrent Managed Templates: Overview	19-1
Concurrent Managed Object Components	19-1
Partition-Level Asynchronous Task Management by CMOs	19-2
Partition-Level MES Template Configuration Elements	19-3
Partition-Level MSES Template Configuration Elements	19-5
Partition-Level MTF Template Configuration Elements	19-5
Partition-Level CMO Constraints for Long-Running Threads	19-6
Setting Limits for Maximum Concurrent Long-Running Requests	19-7
Setting Limits for Maximum Concurrent New Threads	19-9
Configuring Partition CMO Templates: Main Steps	19-9
Using the Administration Console to Configure a Partition-Scoped CMO Template	19-10
Using MBeans to Configure CMO Templates	19-10
Configuring Partition Concurrent Constraints: Main Steps	19-11
Using the Administration Console to Configure Concurrent Constraints for a Partition	19-11
Using MBeans to Configure Concurrent Constraints	19-11
Monitoring Partition-level CMO Templates	19-12
Using the Administration Console to Monitor CMO Templates	19-12
Using MBeans to Monitor CMO Templates	19-12

20 Configuring Partition Batch Job Runtime

Configuring Partition Batch Job Runtime: Overview	20-1
Configuring Partition Batch Job Runtime: Main Steps	20-2
Prerequisites	20-2
Configuring Partition Batch Job Runtime Using the Administration Console	20-3
Configuring Partition Batch Job Runtime: WLST Example	20-3
Querying the Batch Job Runtime	20-4
Using the Administration Console to Query the Batch Job Runtime	20-4
Get Details of All Batch Jobs	20-5
Get Details About a Job's Execution	20-5
Get Details About a Job's Step Execution	20-5
Using Runtime MBeans to Query the Batch Job Runtime	20-6
Get Details of All Batch Jobs Using getJobDetails	20-6
Get Details of a Job Execution Using getJobExecutions	20-7
Get Details of a Job Step Execution Using getStepExecutions	20-8
Configuring Partition Batch Job Runtime: Related Tasks and Links	20-8

21 Configuring Resource Adapters

Configuring Resource Adapters in a Domain Partition: Overview	21-1
Example config.xml for Domain-Level Resource Adapter	21-1
Example config.xml for Partition-Level Resource Adapter	21-2
Best Practices and Considerations When Using Resource Adapters in a Domain Partition	21-3
Defining the Classloading Behavior for Partition-Level Resource Adapters	21-3
Overriding Application Configuration for a Partition	21-3
Considerations for Resource Adapter Resource Definitions	21-4
Resource Group Migration and Resource Adapters	21-5

22 Exporting and Importing Partitions

Exporting and Importing Domain Partitions: Overview	22-1
About Exporting Partitions	22-1
Creating the Partition Archive File	22-2
Partition Archive File Contents	22-2
About Importing Partitions	22-3
Customizing Exporting and Importing System Resources	22-4
Exporting and Importing Applications	22-5
Exporting and Importing Encrypted Attributes	22-5

Exporting and Importing Domain Partitions: Main Steps and WLST Examples	22-6
Exporting Domain Partitions: Main Steps	22-6
Exporting Domain Partitions: WLST Example	22-7
Importing Domain Partitions: Main Steps	22-7
Importing Domain Partitions: WLST Example	22-7
Exporting and Importing Domain Partitions: Related Tasks and Links	22-7

23 Managing Named Concurrent Edit Sessions

Named Concurrent Edit Sessions: Overview	23-1
Managing Named Concurrent Edit Sessions: Main Steps	23-1
Managing Named Concurrent Edit Sessions: WLST Example	23-2
Managing Named Concurrent Edit Sessions: Related Tasks and Links	23-4

24 Configuring Transactions

Configuring Transactions in a Domain Partition: Overview	24-1
Configuring LLR Data Source, JDBC TLog Data Source, and a Determiner Resource in a Domain Partition: Limitations	24-1
Stateful EJB Resource Group Migration Failover: Limitations	24-2

25 Configuring Web Services

Configuring Web Services: Overview	25-1
Configuring Web Services: Main Steps	25-1
Configuring Web Services: Using Partitioned Distributed Topics with SOAP over JMS	25-3
Configuring Web Services: Related Tasks and Links	25-3

26 Monitoring and Debugging Partitions

Monitoring Domain Partitions: Overview	26-1
Partition Log and Diagnostics Data	26-2
Configuring Partition-Scoped Logging	26-3
Configuring Partition-Scoped Debugging	26-4
Configuring Diagnostic System Modules	26-5
Metrics Harvesting	26-5
Configuring Policies and Actions	26-6
Accessing Diagnostic Data	26-6
Shared Log Files	26-7
Partition-Specific Log Files	26-8
Using the WLDF Data Accessor	26-8

Monitoring Resource Consumption Management	26-9
Configuring RCM	26-10
Partition-Scoped RCM Metrics	26-10
Instrumenting Partition-Scoped Applications	26-11
Configuring Partition-Scoped Diagnostic Image Capture	26-12
WLST Diagnostic Commands for Partition Administrators	26-12

Preface

This guide describes the features and use of Oracle WebLogic Server Multitenant and how you can use domain partitions in your WebLogic Server Multitenant (MT) environment.

Audience

This guide is intended for system administrators. Oracle WebLogic Server system administrators have complete access and control over the domain, including applications and resources within domain partitions.

Related administrator responsibilities may include maintaining critical production systems, configuring and managing security realms, implementing authentication and authorization schemes for server and application resources, upgrading security features, and maintaining security provider databases. System administrators have in-depth knowledge of the Java security architecture, including web services, web application and Enterprise JavaBeans (EJB) security, public key security, Secure Sockets Layer (SSL), and Security Assertion Markup Language (SAML).

Documentation Accessibility

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

Access to Oracle Support

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

Related Documents

The following information is relevant to WebLogic Server MT:

- *Administering Oracle WebLogic Server with Fusion Middleware Control* - Describes how to configure, manage, and monitor Oracle WebLogic Server and Oracle WebLogic Server Multitenant using Fusion Middleware Control.
- *Deploying Applications to Oracle WebLogic Server* - Describes deploying Java EE applications or application modules to WebLogic Server instances or clusters, including deploying to domain partitions, resource group templates, and resource groups.

- *Administering Security for Oracle WebLogic Server* - Explains how to configure security for WebLogic Server.
- *Securing Resources Using Roles and Policies for Oracle WebLogic Server* - Introduces the various types of WebLogic Server resources, and provides information that lets you secure these resources using WebLogic Server. The current version of this document primarily focuses on securing URL (web) and Enterprise JavaBeans (EJB) resources.
- *Upgrading Oracle WebLogic Server* - Provides procedures and other information that you need to upgrade 6*n* and earlier versions of WebLogic Server to the latest version. It also provides information about moving applications from a 6*n* or earlier version.

Conventions

The following text conventions are used in this guide:

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.

New and Changed Features in This Release

For a comprehensive listing of the new WebLogic Server features introduced in this release, see *What's New in Oracle WebLogic Server*.

1

Oracle WebLogic Server Multitenant

Oracle WebLogic Server Multitenant (MT) enhances the Oracle platform for Software as a Service (SaaS) and Platform as a Service (PaaS) and provides complete lifecycle management: web tier, middle tier, cache, and database.

Note:

WebLogic Server Multitenant domain partitions, resource groups, resource group templates, virtual targets, and Resource Consumption Management are deprecated in WebLogic Server 12.2.1.4.0 and will be removed in the next release.

WebLogic Server Multitenant domain partitions enable the configuration of a portion of a WebLogic domain that is dedicated to running application instances and related resources. Oracle recommends that customers using domain partitions as a container dedicated to specific applications and resources consider the use of alternative container-based architectures, including the deployment of WebLogic applications and services in Docker containers running in Kubernetes clusters.

The following information introduces WebLogic Server MT and describes its use in multitenant environments.

- [About WebLogic Server MT](#)
- [Key Concepts in WebLogic Server MT](#)
- [WebLogic Server MT Administrators](#)
- [Understanding SaaS Multitenancy](#)
- [Understanding PaaS Multitenancy](#)
- [Lifecycle Management](#)

Note:

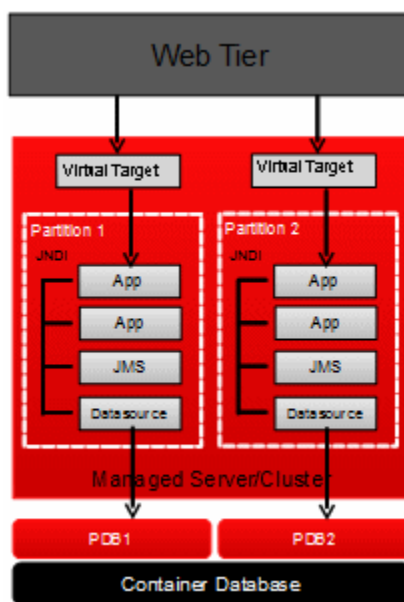
Many new WebLogic Server MT features such as resource groups, resource group templates, and deployment plan overrides are also available in the non-MT version of WebLogic Server. This chapter notes when a given feature is available in the non-MT version and refers to the WebLogic Server documentation and online help where appropriate.

About WebLogic Server MT

Multitenancy in WebLogic Server provides a sharable infrastructure for use by multiple organizations. These organizations are a conceptual grouping of your own choosing, which you can think of as **tenants**. By allowing one domain to support multiple tenants, WebLogic Server MT improves density and achieves a more efficient use of resources while eliminating the hurdles typically present when trying to share multiple applications: runtime cross-application effect, security differences, data commingling, and administrative challenges.

WebLogic Server MT provides resource isolation within a **domain partition**, an administrative and runtime slice of a WebLogic Server domain that is dedicated to running application instances and related resources for a tenant. Domain partitions achieve greater density by allowing application instances and related resources to share the domain, WebLogic Server itself, the Java Virtual Machine (JVM), and the operating system while isolating tenant-specific application data, configuration, and runtime traffic, as shown in [Figure 1-1](#).

Figure 1-1 WebLogic Server Domain Partitions



Each domain partition has its own runtime copy of the applications and resources. Changes in how WebLogic Server handles classloading provide both application isolation and efficiency. Deploying to multitenant environments requires no changes to your applications. For example, you could run multiple instances of a payroll application in different domain partitions without modifying the application.

WebLogic Server MT enables an end to end multitenant infrastructure, including multitenancy from the load balancer to the middle tier and cache tier, and to the database tier. WebLogic Server MT extends the Oracle WebLogic Server Enterprise Edition and Oracle WebLogic Suite products, and includes the following components:

- Oracle WebLogic Server MT, which enables the consolidation of applications into fewer domains (by allowing partitions within domains) while maintaining secure isolation

- WebLogic MT extensions to Java SE Advanced, which enables memory, CPU and I/O isolation, monitoring, and management for applications within a JVM
- Oracle WebLogic Coherence Enterprise Edition to Grid Edition option, which enables the consolidation of caches into fewer Oracle Coherence clusters while maintaining secure isolation
- Oracle Traffic Director, which provides WebLogic Server MT-aware and fully integrated tenant-aware local load balancing

WebLogic Server MT Supports Only Java EE Applications

In this release of WebLogic Server MT, only Java Platform, Enterprise Edition (Java EE) applications are supported. Products that depend on Oracle Java Required Files (JRF) are not supported.

JRF consists of those components not included in the WebLogic Server installation that provide common functionality for Oracle business applications and application frameworks. JRF consists of a number of independently developed libraries and applications that are deployed into a common location. The following components are considered part of JRF: Oracle Application Development Framework, Oracle Fusion Middleware Audit Framework, Dynamic Monitoring Service, Fabric Common, HTTP Client, Infrastructure Security, Java Object Cache, JMX Framework, JPS, logging, MDS, OJSP.Next, Oracle Web Services, Oracle Web Services Manager, Oracle TopLink, UCP, XDK.

This means that WebLogic Server MT does not support the following products:

- Oracle Web Services Manager
- Oracle SOA Suite
- Oracle Application Development Framework (Oracle ADF)
- Oracle WebCenter
- Oracle Service Bus
- Oracle Enterprise Scheduler
- Oracle WebLogic Service Component Architecture (SCA)

Key Concepts in WebLogic Server MT

In addition to domain partitions, WebLogic Server MT introduces these new concepts: tenants, resource groups, resource group templates, virtual targets, scopes, and resource isolation.

The following sections describe their use.

Tenants

Distinct user organizations, such as different external companies (for example, Company A and Company B), or different departments within a single company (for example, HR and Finance), that use applications and resources within a WebLogic Server domain.

A tenant is a logical grouping of your own choosing; it is not a configurable object. That is, you manage domain partitions, not tenants.

A tenant's identity is the association of a given user with a given tenant. For example, you might choose to associate the tenant Finance with a specific domain partition called *Finance-partition*.

The system identifies which tenant a given user belongs to from the user's identity as prescribed by the user identity store. Further, the user's identity helps the system enforce what that user will be authorized to do as they move throughout the system, including but not limited to, which tenant the user belongs to.

Resource Groups

A named collection of (typically) related deployable resources, such as Java EE applications and the data sources, JMS artifacts, and other resources that the applications use.

A traditional WebLogic Server domain may contain many types of deployable resources: Java EE applications, JMS servers and queues, data sources, and such. In this traditional model, if an application suite contains multiple Java EE applications and various resources that support those applications, the administrator defines these resources and deploys these applications individually rather than as a coherent unit.

WebLogic Server MT introduces resource groups, simply as a convenient way to group together Java EE applications and the resources they use into a distinct administrative unit within the domain. The resources and applications are *fully qualified* in that the administrator provides all the information needed to start or connect to those resources, including credentials for connecting to a data source and targeting information for Java EE applications. A resource group will either contain these deployable resources directly or refer to a resource group template that defines the resources. Resource groups can be defined at the domain level, or be specific to a domain partition.

All the resources in or referenced by a resource group are targeted together (to the same target). Resource groups can be started and stopped.

Resource Group Templates

A named, domain-level collection of deployable resources intended to be used as a pattern by (usually) multiple resource groups. Each resource group that refers to a given template will have its own runtime copies of the resources defined in the template. A resource group template is a convenient way to define and replicate resources for multiple tenants. Resource group templates make it very easy to deploy the same collection of applications and resources to multiple domain partitions.

Resource group templates can define:

- Application deployments
- Library deployments
- Java Database Connectivity (JDBC) system resources
- JMS system resources
- Oracle Coherence system resources
- File stores
- JMS servers
- Messaging bridges

- Path services
- Persistent stores
- Store-and-Forward (SAF) agents
- Foreign JNDI providers
- Mail sessions
- WebLogic Diagnostic Framework (WLDF) modules

Resource group templates are defined at the domain level, and then referenced by one or more resource groups.

Resource group templates are particularly useful in a SaaS environment where WebLogic Server MT activates the same applications and resources multiple times, once per domain partition. Some of the information about such resources is the same across all domain partitions, while some of it, such as the attributes of a JMS queue or of a database connection, varies from one partition to the next. For example, the URL, user name, and password used to connect to a data source would be different among different partitions. WebLogic Server MT will activate the application suite represented by the template differently for each partition. You can specify override configuration MBeans and resource deployment plans to override some attribute values in a partition. See [Configuring Resource Overrides](#).

Virtual Targets

The locations where a partition or resource group runs and instructions how to route traffic to them, including addresses, protocol settings, and targeting. Request routing is determined by the host name and optional URI.

May include:

- Host name and port
- Optional URI
- Network access point/channel
- Protocol specific configuration
- Target clusters and Managed Servers

Virtual targets isolate a domain partition and the global (domain-level) runtime from the physical attributes of the target WebLogic Server instances so that your application does not have to know these details.

Scopes

When you deploy an application or library, you have four deployment scope options:

- **Global.** This is the equivalent of the domain level in a nonpartitioned environment.
- **Resource group template.** This is always at the domain level. Whether the application or library that you deploy to a resource group template is available at the domain level or a partition depends on the scope of the resource group that references the resource group template.
- **Resource group in a partition.** This is the only scope that is limited to a partition.
- **Resource group at the domain level.**

You cannot share an application or library between domain partitions: the application or library is available only within the partition. When you deploy the application or library, you specify the resource group in the partition. In Fusion Middleware Control, applications and libraries that are deployed to a resource group in a partition display the name of the domain partition and the resource group within that partition where they are deployed.

The key differences between an application or class running at the domain level and an application or class running in the context of a partition are:

- Applications or classes running at the domain level are available across the domain and are *not* available in partitions.
- Applications or classes running in the context of a partition are available only within that partition.

 **Note:**

The recommended best practice is that you deploy all applications and resources to domain partitions.

Resource Isolation

Resource isolation is critically important within a partitioned environment. When you create a resource group in a domain partition, WebLogic Server MT creates runtime copies of all the resources needed by the application, including JMS servers, JMS stores and JMS modules (including connection factories, queues, topics, and such), Java Connector Architecture (JCA) adapters, and other associated resources.

When you deploy an application or library within a partition, any configuration changes that you make are specific to that partition. WebLogic Server MT ensures that applications in one partition refer to the resources that apply to that partition only and not to resources associated with other partitions.

Resource isolation in WebLogic Server MT encompasses:

- Security: A unique security realm and identity domain for each domain partition.
- Administration:
 - Partition-specific administration
 - Independent deployment, configuration, and software updates
 - Partition-specific monitoring and troubleshooting
- Runtime:
 - Dedicated Java Naming and Directory Interface (JNDI): Isolation between partitions at the JNDI level, which is partition-aware. Applications can bind named objects and retrieve them on a per-partition basis. Each resource is tagged (internally) with a partition-specific name, and is placed in a JNDI tree that is exclusive to the partition. Intra- and inter-application communication for a partition within a cluster is automatically isolated for the partition.
 - Isolated data: JMS, file system, and data sources.

- Resource isolation and fairness: Resource consumption management (RCM) of resources provided by the JVM or the OS. This allows system administrators to specify resource consumption management policies (including constraints, recourse actions and notifications) on resources such as CPU, heap, file, and network.
- Partition Work Managers: The fair distribution of thread usage and prioritization of work requests among partitions that share the same WebLogic Server instance.

WebLogic Server MT Administrators

Two main administrators of WebLogic Server MT are WebLogic Server system administrators, who create and configure partitions in a domain, and partition administrators, who log in directly to a domain partition.

WebLogic Server system administrators create and delete partitions for tenants. System administrators can set or change the security characteristics of a partition (such as the security realm or SSL configuration), or reference a shared (domain-level) resource group or resource group template.

The significant difference between WebLogic Server system administrators and partition administrators is that partition administrators log in directly to a partition-specific security realm, as described in [Administrative Roles for Configuration and Management](#).

Partition administrators manage partitions at the partition level, such as changing partition configuration, deploying applications, creating JMS resources, data sources, JDBC connections, and such, for each resource group in the partition. The partition administrator can also manage the security realm data associated with the partition, such as managing users and groups, credential maps, roles, and policies. To enable security management capabilities for the partition administrator, the system administrator must specify a management identity domain to which the partition administrator belongs. The management identity domain must be the same as the primary identity domain for the partition as shown in Step 5 in [Configuring Security Realms and Primary Identity Domains: Main Steps and Examples](#).

Both system and partition administrators:

- Create, modify, and delete resource groups in partitions.
- Deploy and undeploy applications to resource groups in partitions.
- Start and stop partitions: all of the resource groups and all of the applications deployed to those resource groups are started or shut down.
- Manage users and groups, credential maps, roles, and policies in the partitions.

Analogous to their WebLogic counterparts, partition-constrained roles exist for Deployers, Operators, and Monitors.

Understanding SaaS Multitenancy

The WebLogic Server MT SaaS model provides a way to get the most efficient use from server resources while still providing resource isolation.

In a SaaS environment, if an application cannot internally provide a per-tenant view or the necessary per-tenant isolation, then you might instead deploy separate instances

of the application and its related server-side resources for each tenant. At the very least, this is inefficient: each tenant might get its own stack that includes hardware capacity or Java virtual machines, WebLogic Server domains, Administration Servers, Managed Servers, clusters, and other related resources, such as web servers, data grids, networking, and storage.

With WebLogic Server MT, each tenant gets an application instance plus resource instances on the targeted servers and clusters. The same application or applications run concurrently and independently on multiple domain partitions. No code changes to the applications are needed: WebLogic Server manages the domain partition identification and isolation.

In the SaaS model, you typically define one or more applications and the resources they depend on in a resource group template. You then reference this template from every domain partition in which you want to use the same applications. You make any domain partition-specific changes by editing the values of the associated resource group.

From a high-level view, the SaaS model procedural flow is:

1. The WebLogic Server system administrator creates the JMS resources, data sources, and other resources for a resource group template.
2. The WebLogic Server system administrator deploys the needed applications to the resource group template.
3. When ready to deploy for a tenant, the system administrator does the following:
 - a. Creates a virtual target, if necessary.
 - b. Creates a domain partition configuration, including the resource group, pluggable database information, the virtual target, and the security realm, and then targets the partition to a cluster or set of Managed Servers.
 - c. Overrides some attributes in certain resource groups and all application deployment values.
 - d. Starts the partition, which starts all of the applications deployed to the resource group for that partition.
4. After a partition is started, a partition administrator can view the runtime state of the partition applications and resources, view partition log entries, manage the security realm data in the partition, and stop and restart the partition applications.

At deployment, the applications and resources in the resource group template are replicated and deployed for each domain partition in a resource group, with the following results:

- WebLogic Server infrastructure is shared among domain partitions.
- Separate application instances have their own JNDI tree and resources.
- Runtime traffic is isolated. You access the application using the virtual target.
- Partition Work Managers provide fairness of thread usage and prioritization of work requests among partitions.
- Data is segregated with pluggable databases. The data source implementation creates a separate physical connection pool for each domain partition to its assigned pluggable database.

Testing Applications in a SaaS Environment

One of the many benefits of using domain partitions in a SaaS environment is to test new versions of applications in a controlled production environment.

For example, assume that a business-critical application is deployed to `Partition A`. If you make significant changes to this application, then you would probably want to test it in your production environment before it is generally available to users. WebLogic Server MT provides an easy way to do this. You could create `Partition B` and deploy the updated application, without affecting the stable version of the application running in `Partition A`.

You can target the updated version of the application running in `Partition B` to real production clusters and data sources, test and tune performance metrics, and so forth, all in a controlled environment.

Understanding PaaS Multitenancy

The WebLogic Server MT PaaS model is synonymous with consolidation. Consolidation means that you can deploy different applications from many tenants into the same WebLogic Server infrastructure. WebLogic Server MT shares the infrastructure and underlying resources including the domain, clusters, Managed Servers, hardware, and network.

In the SaaS use case, the WebLogic Server system administrator typically manages all the domain partitions and their resource instances. However, in the PaaS use case, partition administrators are more likely to administer their respective domain partitions and the resources within them, including configuring resources, managing security realm data in the partition, and deploying applications.

Consolidation:

- Makes it easy to deploy applications from many tenants into the same WebLogic Server infrastructure.
- Results in one domain to manage and one JVM consuming resources.
- Isolates management. WebLogic Server system administrators manage the infrastructure. Partition administrators manage partition security realm data, partition deployments and related resources.
- Isolates the runtime, including partition-specific components: security realm (per tenant), virtual target (addresses), pluggable database, JNDI (internal traffic), JMS, Partition Work Managers. Each tenant gets its own set of application instances and dedicated resources.

In the PaaS use case, each tenant would typically run different applications, and these applications may or may not overlap with the applications run by other tenants. When implementing a WebLogic Server MT PaaS solution, you would typically create self-contained resource groups that do not refer to resource group templates but instead represent applications and their related resources that are to be available only within that domain partition. However, even though a PaaS solution is less likely to use resource group templates, this does not mean you cannot use resource group templates in a PaaS solution if they fit your use case.

From a high-level view, the PaaS model procedural flow is:

1. The WebLogic Server system administrator creates a virtual target, if necessary.
2. The WebLogic Server system administrator creates a domain partition with a specified Partition Work Manager or quality of service (QoS) definition to manage the workload.

The Partition Work Manager defines the relative priority and underlying memory and CPU availability for the partition.

3. The WebLogic Server system administrator selects a security realm for the domain partition, and sets the list of available targets.

Resources and applications in the domain partition are available only to users within the domain partition's security realm. Other tenants cannot see or access the resources or applications.

4. The WebLogic Server system administrator assigns the partition to a partition administrator.

Resources, applications, and partition security realm data are managed by the partition administrator. (The underlying Managed Servers and clusters are not managed by a partition administrator.)

5. The partition administrator creates one or more resource groups for that domain partition and selects the resource group from the list of available targets.
6. The partition administrator creates different JMS resources, data sources, PDB info, application deployments, and such for each resource group in the domain partition.
7. The partition administrator starts the partition, which starts all of the applications deployed to the resource groups in the partition.
8. Repeat steps 1-4 for each additional tenant.

Lifecycle Management

Enterprise deployments of Oracle Fusion Middleware can be complex. WebLogic Server MT simplifies consolidating many domains into one using partitions for isolation and introduces new levels of coordination between component configurations.

WebLogic Server MT lifecycle management connects partitions across different components to form one cohesive unit that serves a tenant's needs. To do this, the lifecycle manager (LCM) provides configuration integration across components.

When a partition is created, updated, or deleted—either through Representational State Transfer (REST), WebLogic Scripting Tool (WLST), Fusion Middleware Control, or the Oracle WebLogic Server Administration Console—the lifecycle manager detects these *out of band* actions and synchronizes its configuration with the Administration Server's configuration. The lifecycle manager then informs registered plug-ins, such as the Oracle Traffic Director plug-in, of these changes to keep the entire environment synchronized.

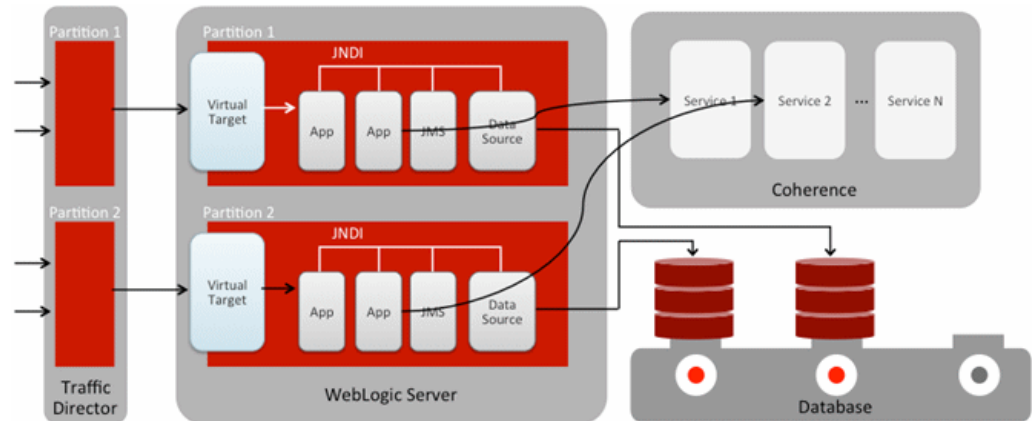
When a domain partition is created or updated in WebLogic Server MT, the changes are orchestrated across multiple tiers:

- The Oracle Traffic Director configuration is updated and synchronized with the new partition.
- Applications are associated with Oracle Coherence caches or services.

- Data sources are associated with existing databases or pluggable databases (PDBs), or new pluggable databases are created.

Figure 1-2 depicts the traffic segregation and flow from the web tier to the database tier.

Figure 1-2 Lifecycle Management



2

Configuring Oracle WebLogic Server Multitenant

Configuring and managing Oracle WebLogic Server Multitenant (MT) may involve several Oracle Fusion Middleware tools and products, including Oracle Fusion Middleware Control, Oracle WebLogic Scripting Tool (WLST), Oracle Coherence, and Oracle Traffic Director.

Note:

WebLogic Server Multitenant domain partitions, resource groups, resource group templates, virtual targets, and Resource Consumption Management are deprecated in WebLogic Server 12.2.1.4.0 and will be removed in the next release.

The following information describes a high-level view of managing and configuring Oracle WebLogic Server Multitenant (MT):

- [How to Manage WebLogic Server MT](#)
- [Configuring WebLogic Server MT](#)

How to Manage WebLogic Server MT

You can use your choice of the following tools to manage WebLogic Server MT: Oracle Enterprise Manager Fusion Middleware Control (FMWC), which is the preferred graphical user interface, Oracle WebLogic Server Administration Console, WebLogic Scripting Tool, and Representational State Transfer (REST) APIs.

For each task, this chapter presents:

- The main steps for accomplishing a given task, with a link to the related Fusion Middleware Control online help topic
- A WLST example

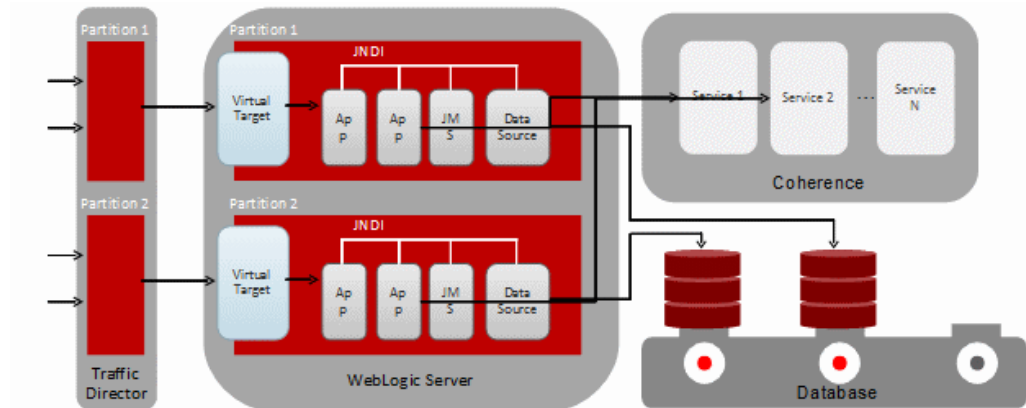
In some instances, there is a difference in how a feature is presented by Fusion Middleware Control and the WebLogic Server Administration Console. This chapter identifies those instances, and includes a link to the WebLogic Server Administration Console online help where appropriate.

Configuring WebLogic Server MT

Configuring a WebLogic Server MT environment encompasses several main steps, including installing WebLogic Server MT and, optionally, Oracle Traffic Director, creating domains, virtual targets, and domain partitions, configuring resource groups, deploying applications, and more.

This high-level view of WebLogic Server MT shows the relationship of domain partitions, Oracle Traffic Director, a WebLogic Server domain, and Oracle Coherence. Consider the graphic shown in [Figure 2-1](#).

Figure 2-1 WebLogic Server MT High-Level View



To configure WebLogic Server MT:

1. Install WebLogic Server for MT, as described in *Installing Oracle WebLogic Server and Oracle Coherence for WebLogic Server MT* in *Installing and Configuring Oracle WebLogic Server and Coherence*.

 **Note:**

If you plan to use Oracle Traffic Director to manage traffic to applications running in partitions, then you must install WebLogic Server in the same path on any remote host where Managed Servers will run. New lifecycle management facilities require access to plug-in Java Archive (JAR) files that must be available at the same relative path as installed on the Administration Server host.

2. To use Oracle Traffic Director to manage traffic to your partitions, install it in collocated mode as described in *Oracle Traffic Director Installation Guide*.
You can install Oracle Traffic Director and WebLogic Server in different `Oracle_Home` locations on different systems (as shown in [Figure 2-1](#)), or in the same `Oracle_Home` location on a single system.
3. To use Oracle Traffic Director to manage traffic to your partitions, use the Configuration Wizard to create an Oracle Traffic Director domain, as described in [Creating the Domain for Oracle Traffic Director](#).

Use the `Oracle Traffic Director - Restricted JRF` template to create the domain. This template automatically includes several other necessary templates.

4. Create a new domain, as described in *Creating a WebLogic Server domain* in *Creating WebLogic Domains Using the Configuration Wizard*.

Use the `Oracle Enterprise Manager - Restricted JRF` template to create the domain, as described in *Installing Oracle WebLogic Server and Coherence for*

WebLogic Server Multitenant in *Installing and Configuring Oracle WebLogic Server and Coherence*. This template automatically includes several other necessary templates.

5. To use the WebLogic Server lifecycle manager (LCM) to coordinate partition configuration changes with Oracle Traffic Director, use WLST to enable the LCM on the WebLogic Server domain:

```
edit()
startEdit()
cd("/")
lcmConfig=cmo.getLifecycleManagerConfig();
lcmConfig.setDeploymentType("admin")
lcmConfig.setOutOfBandEnabled(true)
```

6. Create any clusters and Managed Servers that you want to use for domain partitions.

If you use Fusion Middleware Control or the WebLogic Server Administration Console, then there is nothing partition-specific when creating a cluster. See *Setting up WebLogic Clusters in Administering Clusters for Oracle WebLogic Server*.

However, if you use WLST to create Managed Servers (configured or dynamic), then the required Oracle Java Required Files (JRF) template is not applied. Fusion Middleware Control requires the JRF template to enable domain monitoring.

Therefore, for the WLST use case:

- a. Use WLST to create the cluster or Managed Server.
 - b. Use the `applyJRF` command to apply the JRF template to the Managed Servers. See *WLST Command Reference for Infrastructure Components*.
7. Create one or more virtual targets, as described in [Configuring Virtual Targets](#).
 8. Create and configure a resource group template, as described in [Configuring Resource Group Templates](#).
 9. Optionally, deploy applications to the resource group template, as described in [Deploying Applications](#). Your applications might require partition-specific database connections.
 10. Create a new security realm as described in [Configuring Security](#).
 11. To use Oracle Traffic Director to manage traffic to your partitions, use Fusion Middleware Control to create an Oracle Traffic Director instance, as described in [Creating an Oracle Traffic Director MT Configuration and Instance](#).
 12. Create domain partitions, as described in [Configuring Domain Partitions](#).
 13. Override resources such as Java Database Connectivity (JDBC) connections, as described in [Configuring Resource Overrides](#).
 14. Optionally, configure Oracle Coherence, as described in [Configuring Oracle Coherence](#).
 15. Optionally, configure resource managers, as described in [Configuring Resource Consumption Management](#).
 16. Monitor your domain partitions, as described in [Monitoring and Debugging Partitions](#).

3

Configuring Virtual Targets

Configuring a virtual target includes providing host names to which the virtual target will serve requests and the names of available Managed Servers or a cluster to which the virtual target is itself targeted. A virtual target is a prerequisite when creating a domain partition or a resource group.

 **Note:**

WebLogic Server Multitenant domain partitions, resource groups, resource group templates, virtual targets, and Resource Consumption Management are deprecated in WebLogic Server 12.2.1.4.0 and will be removed in the next release.

This chapter includes the following sections:

- [About Virtual Targets](#)
- [Creating Virtual Targets: Main Steps and Examples](#)
- [Managing Virtual Targets: Main Steps and Examples](#)
- [Configuring Virtual Targets: Related Tasks and Links](#)

About Virtual Targets

A **virtual target** represents a target for a resource group, both at the domain level and in a domain partition. It defines the access points to resources, such as one or more host names, a URI prefix, and the Managed Servers or cluster to which the virtual target is itself targeted.

Virtual targets are similar to virtual hosts in Oracle WebLogic Server. Like virtual hosts, virtual targets provide a separate HTTP server (web container) on each target.

A virtual target isolates a resource group from the physical attributes of where the container is running.

Each resource group requires one or more virtual targets. The virtual targets that you can select depend on whether the resource group is at the domain level, or associated with a domain partition:

- For domain-level resource groups, you can select any existing virtual target that is not already assigned as an available target to a partition.
- For resource groups in a domain partition, you must select only from the existing virtual targets available to the partition.

Consider the following guidelines for using virtual targets:

- A virtual target can be used by only one partition at a time, or at the domain level. A virtual target is never shared by more than one partition, or by a partition and the domain level.
- A virtual target can be used by many resource groups within a partition, or by many resource groups at the domain level.
- A virtual target can be targeted to one cluster or one Managed Server.
- Many virtual targets can be targeted to the same cluster or Managed Server.
- If two virtual targets have the same host names, URI prefix, and port number, they must be targeted to different clusters or Managed Servers.

Components of a Virtual Target

A virtual target contains the following:

- Host names. The list of host names to which this virtual target will serve requests. The host names that you specify must have domain name system (DNS) entries that resolve to the correct servers, or to a load balancer:
 - If you are not using Oracle Traffic Director to balance the connection load for a partition, then specify the actual host name of the WebLogic Server cluster or Managed Server.
 - If you are using Oracle Traffic Director to load balance connections for a partition, then you access applications through Oracle Traffic Director. The host names that you specify for a virtual target must resolve to the location of the Oracle Traffic Director Administration Server.

If you do not specify a host name, then it is the equivalent of using a wild card for the host name to match all incoming requests.

The host name used by a client to access the resource group must exactly match one of the host names specified in the virtual target.

You can specify multiple host names for the virtual target. You might find it convenient to specify both the simple and the fully qualified host name to ensure a match.

- Optional URI prefix to which this virtual target will serve requests. For example, if you enter `www.example.com` as the host name and `MyApp` as the URI prefix, then this virtual target serves requests to `www.example.com/MyApp`, but not to `www.example.com` or `www.example.com/foo`.

To extend this example, assume that your application root is `/app`. The resulting URL for the application would be `www.example.com/MyApp/app`.

- Target cluster and Managed Server from the list of the targets in the current domain. Targets must be either a single server or a single cluster.

There are two general approaches when using virtual targets: route by the host name (`http://partition1.com`, `http://partition2.com`), or by URI prefix (`http://example.com/partition1`, `http://example.com/partition2`). You might find it easier to manage your environment if you consistently use one approach, but there is no requirement that you do so.

You must specify at least one of host name, URI prefix, or port number.

Virtual Targets and Load Balancing

Virtual targets do not determine how a request gets to a particular server. DNS or a load balancer determines how requests get to a specific server instance. The host names used in the virtual target do not control the routing, but they must be consistent with it.

Virtual targets do determine what happens to a request after it enters a server. When a request enters a server, it is matched against all of the virtual targets targeted to that server. If an incoming request matches the information on any of the virtual targets, then it is assumed that the request is intended for that virtual target. If both host names and URI prefix are specified, then both must match. If two virtual targets match, then the one with the longest URI prefix match receives the request. If none matches, then the request is sent to the default HTTP server.

For example, assume that you have a virtual target with the host names `www.example1.com`, `www.example2.com`, and `www.example3.com`. The virtual target is targeted to `cluster1`. Requests sent to `www.example1.com`, `www.example2.com`, and `www.example3.com` all match that virtual target and are all sent to `cluster1`.

If you specify multiple virtual targets for a resource group, then the virtual target matching works the same. There is no implicit load balancing among virtual targets.

For example, assume that in `PartitionA` you have a virtual target with the host name `www.example1.com` targeted to `cluster1`, and a second virtual target with the host name `www.example2.com` targeted to `cluster2`. Requests sent to `www.example1.com` match only that virtual target and are sent only to `cluster1`. Requests sent to `www.example2.com` match only that virtual target and are sent only to `cluster2`.

Use Oracle Traffic Director if you require load balancing.

Using Partition Channels with Virtual Targets

You can optionally specify existing channel and port (explicit or offset) information to use as a reference for a partition-specific channel for a virtual target.

A network channel is a configurable resource that defines the attributes of a network connection to WebLogic Server. A network channel in a WebLogic Server instance is a combination of the following four attributes: communication protocol, listen address, listen port, and channel name. See *Understanding Network Channels in Administering Server Environments for Oracle WebLogic Server*.

The channel must exist on the selected target and be configured using the same protocol. If you specify a value for `Partition Channel`, then you must specify either an explicit port or a port offset.

Allowable values for an explicit port are 1 to 65535, with 1 to 1023 (well-known or system ports) requiring system (or root) process privileges. The port must not already be bound by another process.

Consider the following example:

1. Create a virtual target with the host name `myexample.com` and the URI prefix `/foo`.
2. In the **Partition Channel** field, specify the existing channel `Channel-1` to use as a reference for creating a partition-specific channel.

3. Channel-1 has a listen address and listen port of 127.0.0.1:7002 and an external address and listen port of 127.0.0.1:7002.
4. Specify a port offset of 5. That results in 7002 +5 for the listen port.
5. myexample.com must resolve to 127.0.0.1.
6. To access this virtual target, use myexample.com:7007/foo.
7. If you create another virtual target that also uses Channel-1, then you must use another port offset (or explicit port) or there will be a port conflict.
8. In all cases, specify either an explicit port value to use as-is, or a port offset. Otherwise, there is no port assigned and the channel is not created and used.

You use the Oracle WebLogic Server Administration Console to create a channel. See [Configure custom network channels in the WebLogic Server Administration Console online help](#).

 **Note:**

The partition channel is ignored if you use Oracle Traffic Director to load balance the domain partition.

About Partition Administrative Virtual Targets

A partition administrative virtual target is a virtual target that is created automatically for every partition when the partition is created and has the domain's Administration Server as its physical target.

A partition administrative virtual target contains the following settings:

- Hostname: empty (wildcard)
- uriPrefix: /partitions/<partition-name>

 **Note:**

You can change the uriPrefix, /partitions, to something else. This setting is used for the entire domain. To change it, use:

```
DomainMBean.setPartitionUriSpace(String uriPrefix);
```

However, Oracle recommends leaving this the default setting if possible.

- Target: the Administration Server
- It inherits its other settings from the Administration Server's default HTTP server

You can retrieve the partition administrative virtual target settings from the partition's configuration MBean.

Using WebLogic Scripting Tool (WLST):

```
serverConfig()  
partition = cmo.lookupPartition(partitionName)  
adminVT = partition.getAdminVirtualTarget()
```

The administrative virtual target is not automatically added to the list of available targets for a partition. The system administrator must explicitly add it to the partition's available targets if partition administrators are to be allowed to use it for resource group targeting.

You can use the administrative virtual target for targeting a partition the same as you would any other virtual target; however, the administrative virtual target is read-only and cannot be changed or deleted.

Creating Virtual Targets: Main Steps and Examples

Creating virtual targets includes certain required steps, such as providing a name, specifying host names, and selecting available targets.

Note:

Do not configure a virtual target as host name: "", URI prefix: /, explicit port: 0, port offset: 0. This configuration makes the virtual target intercept all requests intended for the default HTTP server. This configuration is not allowed.

The main steps for creating a virtual target are as follows:

1. Enter a name for this virtual target.
2. Optionally, enter a URI prefix for this virtual target.
3. Add one or more host names for the virtual target. The host names that you specify must resolve to the target cluster and Managed Servers that you specify, or resolve to the Oracle Traffic Director Administration Server if you are using Oracle Traffic Director for load balancing.
4. Optionally, specify partition channel and port (explicit or offset) information to use for this virtual target, as described in [Using Partition Channels with Virtual Targets](#). The channel must exist on the selected target and be configured using the same protocol. If you specify a value for Partition Channel, then you must specify either an explicit port or a port offset.

Note:

These options are ignored if you use Oracle Traffic Director to load balance the domain partition.

5. Select one target from the available targets. You can select a single Managed Server or a single cluster.
6. If you are using the WebLogic Server Administration Console, WebLogic Scripting Tool (WLST), or Representational State Transfer (REST) to create the virtual target, additional HTTP configuration options are available for configuring the virtual web server. See *Configure HTTP for a virtual target* in *Oracle WebLogic Server Administration Console Online Help*.

To create a virtual target using Fusion Middleware Control, see [Create virtual targets](#) in the online help.

Creating Virtual Targets: WLST Example

The example does the following:

1. Creates a domain partition.
2. Creates a virtual target.
3. Sets the host name and URI prefix for the virtual target.
4. Targets the virtual target to the Administration Server. (You would typically not select the WebLogic Administration Server.)
5. Adds the virtual target as an available target in the partition.
6. Creates the resource group.
7. Adds the virtual target to the resource group.
8. Activates the changes.
9. Starts the partition.

 **Note:**

If this is the first partition created in production mode, then you must restart the Administration Server before you can start the partition.

```
# Create virtual target
edit()
startEdit()
wls:/base_domain/edit/ !> domain=getMBean('/')
wls:/base_domain/edit/ !> peppart=domain.createPartition('Pep')
wls:/base_domain/edit/ !> vt=domain.createVirtualTarget('TestVT')
wls:/base_domain/edit/ !> vt.setHostNames(jarray.array([String('localhost')],String))
wls:/base_domain/edit/ !> vt.setUriPrefix('/foo')
wls:/base_domain/edit/ !> tgt=getMBean('/Servers/AdminServer')
wls:/base_domain/edit/ !> vt.addTarget(tgt)
wls:/base_domain/edit/ !> peppart.addAvailableTarget(vt)
wls:/base_domain/edit/ !> peprg=peppart.createResourceGroup('TestRG')
wls:/base_domain/edit/ !> peprg.addTarget(vt)
wls:/base_domain/edit/ !> activate()
wls:/base_domain/edit/ !> startPartitionWait(peppart)
```

Creating Virtual Targets: REST Example

For an example of creating virtual targets using REST, see [Creating Partitions in Administering Oracle WebLogic Server with RESTful Management Services](#).

In particular, see the section "Create a virtual target for the new partition."

Managing Virtual Targets: Main Steps and Examples

To manage virtual targets, you can modify the URI prefix, host names for the virtual target, and its target servers or cluster.

If you make a change to a virtual target that is in use by a running resource group in a running partition, then you may need to restart the partition to have the change take effect.

The main steps for managing a virtual target are as follows:

1. Select the virtual target that you want to manage.
2. Enter a URI prefix for this virtual target or modify the existing URI prefix.
3. Add one or more host names for the virtual target. The host names that you specify must resolve to the target cluster and Managed Servers that you specify, or resolve to the Oracle Traffic Director Administration Server if you are using Oracle Traffic Director for load balancing.
4. If needed, delete one or more host names for the virtual target.
5. Optionally, specify partition channel and port (explicit or offset) information to use for this virtual target, as described in [Using Partition Channels with Virtual Targets](#). The channel must exist on the selected target and be configured using the same protocol. If you specify a value for Partition Channel, then you must specify either an explicit port or a port offset.

 **Note:**

These options are ignored if you use Oracle Traffic Director to load balance the domain partition.

6. Select one target from the available targets. You can select a single Managed Server or a single cluster. (You would typically not select the WebLogic Server Administration Server unless you have a special reason to do so.)
Deselect any target that you do not want to use.
7. If you are using the WebLogic Server Administration Console, WLST, or REST to manage the virtual target, then additional HTTP configuration options are available for configuring the virtual web server. See *Configure HTTP for a virtual target* in *Oracle WebLogic Server Administration Console Online Help* for more information.
8. If you made a change to a virtual target that is in use by a running resource group in a running partition, then restart the partition to have the change take effect.

To manage a virtual target using Fusion Middleware Control, see *Configure virtual targets* in the Fusion Middleware Control online help.

Deleting Virtual Targets in Use by a Partition

You cannot delete a virtual target that is in use by a resource group in a partition. You must first remove it from the resource group.

To delete a virtual target that is in use by a partition:

1. Stop the partition.
2. Remove the virtual target from the resource group in that partition.
3. Remove the virtual target from the list of available targets for the partition.
4. If the virtual target is used by default with the partition, then deselect this option.

5. Restart the partition.
6. Delete the virtual target.

To delete a virtual target using Fusion Middleware Control, see [Delete virtual targets in use by a partition in the Fusion Middleware Control online help](#).

Deleting Virtual Targets in Use by a Resource Group at the Domain Level

You cannot delete a virtual target that is in use by a resource group at the domain level. You must first remove it from the resource group.

To delete a virtual target that is in use by a resource group at the domain level:

1. Remove the virtual target that you want to delete from the resource group.
2. Delete the virtual target.

To delete a virtual target, see [Delete virtual targets in use by a resource group at the domain level in the Fusion Middleware Control online help](#).

Managing Virtual Targets: WLST Example

The following example modifies the existing virtual target `TestVT` with the URI prefix `/app` and sets the host name to be `myhost.example.com`.

`myVT` is an instance of [VirtualTargetMBean](#).

```
cd('/')
edit()
startEdit()
cd('VirtualTargets')
cd('TestVT')
cmo.setUriPrefix('/app')
cmo.setHostNames(jarray.array([String("myhost.example.com")],String))
activate()
```

You can also use the [DomainMBean](#) to look up a virtual target by name and return the [VirtualTargetMBean](#).

```
domain=getMBean('/')
vt=domain.lookupVirtualTarget('TestVT')
vt.setUriPrefix('/app')
```

Delete a Virtual Target

The following example deletes the virtual target `TestVT`. It first removes the virtual target as an available target for the partition `Pep`.

```
cd('/')
edit()
startEdit()
domain=getMBean('/')
vt=domain.lookupVirtualTarget('TestVT')
part=domain.lookupPartition('Pep')
part.removeAvailableTarget(vt)
domain.destroyVirtualTarget(vt)
activate()
```

Retarget a Virtual Target

The example does the following:

1. Creates a domain partition.
2. Creates a virtual target.
3. Sets the host name and URI prefix for the virtual target.
4. Targets the virtual target to the Administration Server. (You would typically not select the WebLogic Server Administration Server unless you have a special reason to do so.)
5. Adds the virtual target as an available target in the partition.
6. Creates the resource group.
7. Adds the virtual target to the resource group.
8. Activates the changes.
9. Starts the partition.

 **Note:**

If this is the first partition created in production mode, then you must restart the Administration Server before you can start the partition.

10. Removes the Administration Server as a target.

11. Retargets the virtual target to Cluster-0.

```
# Create Pep partition and ResourceGroup
edit()
startEdit()
wls:/base_domain/edit/ !> domain=getMBean('/')
wls:/base_domain/edit/ !> peppart=domain.createPartition('Pep')
wls:/base_domain/edit/ !> vt=domain.createVirtualTarget('TestVT')
wls:/base_domain/edit/ !> vt.setHostNames(jarray.array([String('localhost')],String))
wls:/base_domain/edit/ !> vt.setUriPrefix('/foo')
wls:/base_domain/edit/ !> tgt=getMBean('/Servers/AdminServer')
wls:/base_domain/edit/ !> vt.addTarget(tgt)
wls:/base_domain/edit/ !> peppart.addAvailableTarget(vt)
wls:/base_domain/edit/ !> peprg=peppart.createResourceGroup('TestRG')
wls:/base_domain/edit/ !> peprg.addTarget(vt)
wls:/base_domain/edit/ !> activate()
wls:/base_domain/edit/ !> startPartitionWait(peppart)
startEdit()
wls:/base_domain/edit/ !> vt.removeTarget(tgt)
wls:/base_domain/edit/ !> tgt=getMBean('/Clusters/Cluster-0')
wls:/base_domain/edit/ !> vt.addTarget(tgt)
wls:/base_domain/edit/ !> activate()
```

Managing Virtual Targets: REST Example

For an example of creating virtual targets using REST, see *Creating Partitions in Administering Oracle WebLogic Server with RESTful Management Services*.

In particular, see the section "View the default values for a new virtual target."

Configuring Virtual Targets: Related Tasks and Links

- [Configuring Domain Partitions](#)
- Understanding Network Channels in *Administering Server Environments for Oracle WebLogic Server*

4

Configuring Resource Group Templates

Resource group templates are particularly useful in a Software-as-a-Service environment where WebLogic Server Multitenant (MT) activates the same applications and resources multiple times, once per domain partition.

Note:

WebLogic Server Multitenant domain partitions, resource groups, resource group templates, virtual targets, and Resource Consumption Management are deprecated in WebLogic Server 12.2.1.4.0 and will be removed in the next release.

This chapter includes the following sections:

- [About Resource Group Templates](#)
- [Creating Resource Group Templates: Main Steps and WLST Examples](#)
- [Configuring Resource Group Templates: Main Steps and WLST Examples](#)
- [Deleting Resource Group Templates: Main Steps and WLST Examples](#)
- [Configuring Resource Group Templates: Related Tasks and Links](#)

About Resource Group Templates

A resource group template is a named, domain-level collection of deployable resources intended to be used as a pattern by multiple resource groups. Each resource group that refers to (references) a given template will have its own runtime copies of the resources defined in the template.

A resource group template is a convenient way to define and replicate resources for multiple tenants. Resource group templates make it very easy to deploy the same collection of applications and resources to multiple domain partitions. Some of the information about such resources is the same across all domain partitions, while some of it, such as the attributes of a Java Message Service (JMS) queue or of a database connection, varies from one partition to another.

You can create a resource group template in two ways:

- Create a new resource group template. This creates the basic structure for the resource group template. You must then edit this resource group template as needed.
- If you are using the Oracle WebLogic Server Administration Console, then you can clone (copy) an existing resource group template. The configuration is copied from the template to the new resource group template. You must then edit and override values from the resource group template as needed.

For more information about resource group templates, see the following topics:

- [What Is the Difference Between a Resource Group Template and a Resource Group?](#)
- [What Is in a Resource Group Template?](#)
- [Resource Group Templates and Overrides](#)

What Is the Difference Between a Resource Group Template and a Resource Group?

One major difference between resource group templates and resource groups is that resource group templates do not have targets. Because resource group templates are intended to be used only as a pattern by multiple resource groups, possibly in many partitions, any target-specific information will most likely be unique to the resource group.

Therefore:

- If you deploy applications to the resource group template, then you cannot start the applications until the resource group template is referenced by a resource group.
- You cannot start or stop a resource group template.

What Is in a Resource Group Template?

Resource group templates are based on the [ResourceGroupTemplateMBean](#) and can include the following resources:

- General (name)
- Deployments
- Services
 - Java Database Connectivity (JDBC)
 - Messaging
 - Mail sessions
 - Persistent stores
 - Foreign Java Naming and Directory (JNDI) providers
 - OSGi frameworks
 - Diagnostics
- Notes

Resource Group Templates and Overrides

Overriding resources lets you customize resources at the partition level.

You override resource settings that are derived from a resource group template as follows:

1. Create the resource group template.
2. Deploy applications to the resource group template as needed.

3. Define services in the resource group template as needed.
4. Create a partition and then create a resource group that references the resource group template.
5. Override the values from the resource group template in the resource group.

You can override resource definitions in partitions using override configuration MBeans, resource deployment plans, and partition-specific application deployment plans. For more information about resource overrides, see [Configuring Resource Overrides](#). For information about application overrides, see [Deploying Applications to Partition Resource Groups](#).

Resource Group Template Example

Assume that you have a SaaS environment, and that you want to run an instance of a specific business application in three different partitions. This business application requires a pluggable database, which means you need one pluggable database per partition because each application instance needs its own data isolated from the other application instances.

Assume further that the configuration for the pluggable databases will be identical for the three partitions, with the exception of the database name, user name, and password. For these values, plan to use the values from [Table 4-1](#).

Table 4-1 Example Pluggable Databases

PDB Name	User name/Password	For Use by
pdb1.example.com	pduser1/pduser1	Partition1
pdb2.example.com	pduser2/pduser2	Partition2
pdb3.example.com	pduser3/pduser3	Partition3

To configure your resource group template, follow these steps:

1. Create a resource group template.
2. Go to **Services > JDBC**.
3. Create a JDBC system data source. For the purpose of this example, create a generic data source.
4. Enter the configuration values for the JDBC data source.
5. For the connection properties, use the following values:

You will override these values in the resource groups that reference this resource group template. However, by entering initial values here you can test the database connection.

- Enter the database name. For example, from [Table 4-1](#) enter **pdb1.example.com**.
 - Enter the database user name. For example, from [Table 4-1](#) enter **pduser1**.
 - Enter the password. For example, from [Table 4-1](#) enter **pduser1**.
6. Create your three partitions from [Table 4-1](#), with resource groups based on the resource group template.

7. For `Partition1`, you do not have to override any of the JDBC data source values because you used the correct values in the resource group template.
8. For `Partition2` and `Partition3`, override the database name, user name, and password with the correct values from [Table 4-1](#).

To do this using Oracle Enterprise Manager Fusion Middleware Control, go to **Domain Partitions > Partition > Resource Overrides**, as described in WebLogic Server Resource Overrides in *Administering Oracle WebLogic Server with Fusion Middleware Control*.

Creating Resource Group Templates: Main Steps and WLST Examples

Creating a new resource group template provides only its basic structure. After you create a resource group template, you must configure its resources.

To create a new resource group template:

1. Go to **Domain > Environment > Resource Group Templates**.
2. Create a new resource group template. The name must be unique in the domain.

These tasks are described in Create resource group templates in *Administering Oracle WebLogic Server with Fusion Middleware Control*.

For information about configuring resource group templates, see [Configuring Resource Group Templates: Main Steps and WLST Examples](#).

Creating Resource Groups: WLST Example

The example does the following:

1. Creates a domain partition.
2. Creates a virtual target.
3. Sets the host name and URI prefix for the virtual target.
4. Adds the virtual target as an available target in the partition.
5. Creates the resource group.
6. Adds the virtual target to the resource group.
7. Creates a resource group template.
8. Associates the resource group with the resource group template.
9. Activates the changes.
10. Starts the partition.

 **Note:**

If this is the first partition created in production mode, then you must restart the Administration Server before you can start the partition.

```

# Create Pep partition resource group, and resource group template
edit()
startEdit()
wls:/base_domain/edit/ !> domain=getMBean('/')
wls:/base_domain/edit/ !> peppart=domain.createPartition('Pep')
wls:/base_domain/edit/ !> vt=domain.createVirtualTarget('VirtualTarget-0')
wls:/base_domain/edit/ !> vt.setHostNames(jarray.array([String('localhost')],String))
wls:/base_domain/edit/ !> vt.setUriPrefix('/foo')
wls:/base_domain/edit/ !> peppart.addAvailableTarget(vt)
wls:/base_domain/edit/ !> peprg=peppart.createResourceGroup('TestRG')
wls:/base_domain/edit/ !> peprg.addTarget(vt)
wls:/base_domain/edit/ !> peprgt=domain.createResourceGroupTemplate('TestRGT')
wls:/base_domain/edit/ !> peprg.setResourceGroupTemplate(peprgt)
wls:/base_domain/edit/ !> activate()
wls:/base_domain/edit/ !> startPartitionWait(peppart)

```

Configuring Resource Group Templates: Main Steps and WLST Examples

The initial resource group template configuration is a basic skeleton that you must configure before you can use it. The tasks may include configuring JDBC data sources, JMS servers and resources, foreign JNDI providers, and so forth.

For detailed information about configuring resource group templates, see the following tasks:

- [Configuring Resource Group Template Deployment Settings](#)
- [Configuring Resource Group Template Services Settings](#)
- [Configuring Resource Group Template Notes](#)

Configuring Resource Group Template Deployment Settings

To view and define resource group template deployment settings:

1. Select the resource group template that you want to configure.
2. You can take the following deployment actions:
 - Deploy
 - Redeploy
 - Undeploy
 - Fetch deployment plan
3. Save your changes.

These tasks are described in *Configure resource group template deployments in Administering Oracle WebLogic Server with Fusion Middleware Control*.

See [Deploying Applications to Resource Group Templates](#).

Configuring Resource Group Template Services Settings

This section includes the following tasks:

- [Configuring Resource Group Template JDBC Settings](#)

- [Configuring Resource Group Template JMS settings](#)
- [Configuring Resource Group Mail Session Settings](#)
- [Configuring Resource Group Persistent Store Settings](#)
- [Configuring Resource Group Foreign JNDI Provider Settings](#)
- [Configuring Resource Group Diagnostic System Module Settings](#)

Configuring Resource Group Template JDBC Settings

To view configuration settings for the JDBC system resources that have been created in this resource group template:

1. Select the resource group template that you want to configure.
2. Go to **Services > JDBC**.

The read-only JDBC information for a resource group includes:

- Name
 - JNDI name
 - Type
 - Algorithm type
 - Row prefetch enabled
 - Row prefetch size
 - Stream chunk size
3. Create or delete the system data sources as needed.
 4. Save your changes.

See [Configuring JDBC](#).

Configuring Resource Group Template JMS settings

This section includes the following tasks:

- [Configuring JMS Server Settings](#)
- [Configuring SAF Agent Settings](#)
- [Configuring JMS Resource Settings](#)
- [Configuring JMS Module Settings](#)
- [Configuring Messaging Bridges](#)
- [Configuring JMS Bridge Destinations](#)
- [Configuring Path Services](#)

Configuring JMS Server Settings

To view configuration settings for the JMS servers that have been created for this resource group template:

1. Select the resource group that you want to configure.
2. Go to **Services > Messaging > JMS Servers**.

The following read-only information is available for JMS servers already configured in this resource group template:

- Name
 - Health
 - Health reason
 - Persistent store
 - Temporary template name
 - Bytes maximum
 - Messages maximum
 - Bytes threshold high
 - Bytes threshold low
 - Messages threshold high
3. Create or delete JMS servers as needed.
 4. Save your changes.

These tasks are described in WebLogic Server Messaging in *Administering Oracle WebLogic Server with Fusion Middleware Control*.

See [Configuring Messaging](#).

Configuring SAF Agent Settings

To view configuration settings for the store-and-forward (SAF) agents that have been created for this resource group template:

1. Select the resource group template that you want to configure.
2. Go to **Services > Messaging > SAF Agents**.

The following read-only information is available for the SAF agents configured in this resource group template:

- Name
 - Agent type
 - Persistent store
3. Create or delete SAF agents as needed.
 4. Save your changes.

These tasks are described in Configure SAF Agent Settings in *Administering Oracle WebLogic Server with Fusion Middleware Control*.

See [Configuring Messaging](#).

Configuring JMS Resource Settings

To monitor the resource settings for a resource group template:

1. Select the resource group template that you want to configure.
2. Go to **Services > Messaging > JMS Resources**.

The following read-only information is available for the JMS resources configured in this resource group template:

- Name
 - Type
 - JMS module name
 - JNDI name
 - Subdeployment
3. Create or delete JMS resources as needed.
 4. Save your changes.

These tasks are described in Configure SAF Agent Settings in *Administering Oracle WebLogic Server with Fusion Middleware Control*.

For information about configuring existing JMS resources, see [Configuring Messaging](#).

Configuring JMS Module Settings

To configure the JMS modules for a resource group template:

1. Select the resource group template that you want to configure.
2. Go to **Services > Messaging > JMS Modules > JMS Modules**.

The following read-only information is available for the JMS modules configured in this resource group template:

- Name
 - Queues
 - Topics
 - Connection factories
 - Distributed queues
 - Distributed topics
 - Foreign servers
 - Quotas
 - SAF error handlers
 - SAF imported destinations
 - SAF remote contexts
 - Templates
 - Uniform distributed queues
 - Uniform distributed topics
 - Destination keys
 - Type
3. Create or delete JMS modules as needed.
 4. Save your changes.

These tasks are described in Configure JMS module settings in *Administering Oracle WebLogic Server with Fusion Middleware Control*.

See [Configuring Messaging](#).

Configuring Messaging Bridges

To configure the messaging bridge settings for a resource group template:

1. Select the resource group template that you want to configure.
2. Go to **Services > Messaging > Messaging Bridges**.

The following read-only information is available for the messaging bridges configured in this resource group template:

- Name
 - Source bridge destination
 - Target bridge destination
3. Create or delete messaging bridges as needed.
 4. Save your changes.

These tasks are described in Configure messaging bridges in *Administering Oracle WebLogic Server with Fusion Middleware Control*.

See [Configuring Messaging](#).

Configuring JMS Bridge Destinations

To configure the JMS bridge destination settings for a resource group template:

1. Select the resource group template that you want to configure.
2. Go to **Services > Messaging > Bridge Destinations**.

The following read-only information is available for the JMS bridge destinations configured in this resource group template:

- Name
 - Adapter JNDI name
3. Create or delete JMS bridge destinations as needed.
 4. Save your changes.

These tasks are described in Configure JMS bridge destinations in *Administering Oracle WebLogic Server with Fusion Middleware Control*.

See [Configuring Messaging](#).

Configuring Path Services

To configure the path services settings for a resource group template:

1. Select the resource group template that you want to configure.
2. Go to **Services > Messaging > Path Services**.

The following read-only information is available for the path services configured in this resource group template:

- Name
 - Persistent store
3. Create or delete path services as needed.
 4. Save your changes.

These tasks are described in Configure path services in *Administering Oracle WebLogic Server with Fusion Middleware Control*.

See [Configuring Messaging](#).

Configuring Resource Group Mail Session Settings

To view configuration settings for the mail sessions that have been created in this resource group template:

1. Select the resource group template that you want to configure.
2. Go to **Services > Mail**.

The following read-only information is available for mail sessions configured in this resource group template:

- Name
 - JNDI name
3. Create or delete mail sessions as needed.
 4. Save your changes.

These tasks are described in WebLogic Server Mail Sessions in *Administering Oracle WebLogic Server with Fusion Middleware Control*.

Configuring Resource Group Persistent Store Settings

To view configuration settings for the persistent stores that have been created in this resource group template:

1. Select the resource group template that you want to configure.
2. Go to **Services > Persistent Stores**.

The following read-only information is available for the persistent stores configured in this resource group template:

- Name
 - Type
3. Create or delete persistent stores as needed.
 4. Save your changes.

These tasks are described in WebLogic Server Persistent Stores in *Administering Oracle WebLogic Server with Fusion Middleware Control*.

See [Configuring Messaging](#).

Configuring Resource Group Foreign JNDI Provider Settings

To view configuration settings for the foreign JNDI providers that have been created in this resource group template:

1. Select the resource group template that you want to configure.
2. Go to **Services > Foreign JNDI Providers**.

The following read-only information is available for the foreign JNDI providers configured in this resource group template:

- Name
 - Initial context factory
 - Provider URL
 - User
 - Targets
3. Create or delete foreign JNDI providers as needed.
 4. Save your changes.

These tasks are described in WebLogic Server Foreign JNDI Providers in *Administering Oracle WebLogic Server with Fusion Middleware Control*.

See [Configuring and Programming JNDI](#).

Configuring Resource Group Diagnostic System Module Settings

To view configuration settings for the diagnostic system modules that have been created in this resource group template:

1. Select the resource group template that you want to configure.
2. Go to **Services > Diagnostics**.

The following read-only information is available for the diagnostic system modules configured in this resource group template:

- Name
 - Description
 - Targets
3. Create or delete diagnostic system modules as needed.
 4. Save your changes.

These tasks are described in WebLogic Server Diagnostics in *Administering Oracle WebLogic Server with Fusion Middleware Control*.

See [Monitoring and Debugging Partitions](#).

Configuring Resource Group Template Notes

To create notes for a resource group template:

1. Select the resource group that you want to configure.
2. Go to **Notes**.
3. Enter your notes.
4. Save your changes.

Configuring Resource Group Template: WLST Example

The example does the following:

1. Creates a domain partition.
2. Creates a virtual target.
3. Sets the host name and URI prefix for the virtual target.
4. Adds the virtual target as an available target in the partition.
5. Creates the resource group.
6. Adds the virtual target to the resource group.
7. Creates a resource group template.
8. Deploys the application `MySimpleEjb` to the resource group template.
9. Associates the resource group with the resource group template.
10. Activates the changes.
11. Starts the partition.

Note:

If this is the first partition created in production mode, then you must restart the Administration Server before you can start the partition.

```
# Create Pep partition resource group, and resource group template
edit()
startEdit()
wls:/base_domain/edit/ !> domain=getMBean('/')
wls:/base_domain/edit/ !> peppart=domain.createPartition('Pep')
wls:/base_domain/edit/ !> vt=domain.createVirtualTarget('VirtualTarget-0')
wls:/base_domain/edit/ !> vt.setHostNames(jarray.array([String('localhost')],String))
wls:/base_domain/edit/ !> vt.setUriPrefix('/foo')
wls:/base_domain/edit/ !> peppart.addAvailableTarget(vt)
wls:/base_domain/edit/ !> peprg=peppart.createResourceGroup('TestRG')
wls:/base_domain/edit/ !> peprg.addTarget(vt)
wls:/base_domain/edit/ !> peprgt=domain.createResourceGroupTemplate('TestRGT')
wls:/base_domain/edit/ !> deploy(appName='MySimpleEJB', path='c:/webservices/
MySimpleEjb.jar', resourceGroupTemplate='TestRGT')
wls:/base_domain/edit/ !> peprg.setResourceGroupTemplate(peprgt)
wls:/base_domain/edit/ !> activate()
wls:/base_domain/edit/ !> startPartitionWait(peppart)
```

Deleting Resource Group Templates: Main Steps and WLST Examples

Deleting a resource group template permanently removes it from the domain's configuration. When you delete a resource group template, all of the applications deployed to the resource group template are undeployed. You cannot delete a resource group template that is being referenced by one or more resource groups.

You must first remove the references to the resource group template before you can delete it.

1. Select the referencing resource groups of the resource group template that you want to delete.
2. Delete the resource groups.
3. Delete the resource group template.

Deleting Resource Group Templates: WLST Example

The example does the following:

1. Creates a domain partition.
2. Creates a virtual target.
3. Sets the host name and URI prefix for the virtual target.
4. Adds the virtual target as an available target in the partition.
5. Creates the resource group.
6. Adds the virtual target to the resource group.
7. Create a resource group template.
8. Associates the resource group with the resource group template.
9. Unsets the resource group template from the resource group.
10. Deletes the resource group template from the domain.
11. Activates the changes.
12. Starts the partition.

Note:

If this is the first partition created in production mode, then you must restart the Administration Server before you can start the partition.

```
# Create Pep partition resource group, and resource group template
edit()
startEdit()
wls:/base_domain/edit/ !> domain=getMBean('/')
wls:/base_domain/edit/ !> peppart=domain.createPartition('Pep')
wls:/base_domain/edit/ !> vt=domain.createVirtualTarget('VirtualTarget-0')
wls:/base_domain/edit/ !> vt.setHostNames(jarray.array([String('localhost')],String))
wls:/base_domain/edit/ !> vt.setUriPrefix('/foo')
wls:/base_domain/edit/ !> peppart.addAvailableTarget(vt)
wls:/base_domain/edit/ !> peprg=peppart.createResourceGroup('TestRG')
wls:/base_domain/edit/ !> peprg.addTarget(vt)
wls:/base_domain/edit/ !> peprgt=domain.createResourceGroupTemplate('TestRGT')
wls:/base_domain/edit/ !> peprg.setResourceGroupTemplate(peprgt)
wls:/base_domain/edit/ !> peprg.unSet('ResourceGroupTemplate')
wls:/base_domain/edit/ !> domain.destroyResourceGroupTemplate(peprgt)
wls:/base_domain/edit/ !> activate()
wls:/base_domain/edit/ !> startPartitionWait(peppart)
```

Configuring Resource Group Templates: Related Tasks and Links

- [Resource Group Templates](#)
- [Configuring Resource Groups](#)
- [Configuring Virtual Targets](#)
- [Configuring Messaging](#)

5

Configuring Security

WebLogic Server MT expands upon the traditional WebLogic Server security support in significant ways, with multiple security realms, identity domains, and administrative roles for configuration and management when logged in directly to a partition.

 **Note:**

WebLogic Server Multitenant domain partitions, resource groups, resource group templates, virtual targets, and Resource Consumption Management are deprecated in WebLogic Server 12.2.1.4.0 and will be removed in the next release.

This chapter assumes that you have a working knowledge of the WebLogic Server security concepts and tasks described in *Understanding Security for Oracle WebLogic Server*.

This chapter includes the following sections:

- [New Security Features in Domain Partitions](#)
- [Configuring the Administrative Identity Domain: Main Steps and WLST Example](#)
- [Configuring Security Realms and Primary Identity Domains: Main Steps and Examples](#)
- [Setting the Identity Domain-Aware Providers Required Control: Main Steps](#)
- [Connecting Directly to Domain Partitions: Main Steps and WLST Example](#)
- [Managing Security Data as a Partition Administrator: Main Steps and Examples](#)
- [Configuring SSL in Domain Partitions](#)
- [Configuring Security in Domain Partitions: Related Tasks and Links](#)

New Security Features in Domain Partitions

WebLogic Server MT provides a unique security realm and identity domain for each domain partition, and administrative roles for configuring and managing users and groups, credential maps, roles, and policies in the partitions.

 **Note:**

WebLogic Server Multitenant domain partitions are deprecated in WebLogic Server 12.2.1.4.

- Multiple realms. WebLogic Server MT supports multiple active realms and allows each partition to execute against a different realm.

As before, there is only one active default realm used at the global (domain) level.

- Identity domains. An identity domain is a logical namespace for users and groups, typically representing a discrete set of users and groups in the physical data store. Identity domains are used to identify the users associated with particular partitions.
- Administrative roles for configuration and management when logged in directly to a partition. You can use Fusion Middleware Control and WebLogic Scripting Tool (WLST) to connect directly to a domain partition instead of logging in to the domain level.

These changes are described in the sections that follow.

Domain Partition Security Realms: Overview

As described in *Security Realms in Understanding Security for Oracle WebLogic Server*, a security realm comprises mechanisms for protecting WebLogic Server resources. Each security realm consists of a set of configured security providers, users, groups, security roles, and security policies. You use realms to configure authentication, authorization, role mapping, credential mapping, auditing, and other services.

WebLogic Server traditionally supports multiple realms in a domain configuration, but only one realm—typically referred to as the *default realm* or *admin realm*—can be active at any given time.

In contrast, WebLogic Server MT supports multiple active realms and allows each partition to execute against a different realm.

This means that a partition can have unique security providers, users, groups, security roles, and security policies. Resources and applications in the domain partition are available only to users within the domain partition's security realm. Other tenants cannot see or access the resources or applications.

Note:

Partitions can share a security realm, with consequent loss of independence and isolation. In particular, if you do not specify a realm when you create a partition, then the default realm is shared with the partition and there is no security isolation between the partition and the domain.

You cannot use the RDBMS Authentication providers in a multitenant environment. These providers require a data source that is defined at the domain level and do not support partition-level data sources.

Identity Domains: Overview

Note:

Identity domains are an optional feature you should use only if your user store supports them. The structure, configuration, and management of the identity domain itself are outside the control of, and are invisible to, WebLogic Server.

The purpose of an identity store is for a partition's users to be able to access the partition, while other partitions' users cannot.

Identity domains distinguish users associated with different partitions. You can configure a partition with an identity domain that identifies the set of users associated with the partition. Access policies allow those users—but not users from other identity domains—to access the partition.

Identity domain names should be meaningful in your environment.

There is a generally a one-to-one mapping between partitions and identity domains. However, it is possible for multiple partitions to use the same identity domain. The result is to remove any distinction between users of those partitions, which may not be appropriate in your environment.

Types of Identity Domains

There are two types of identity domains:

- **Administrative identity domain:** The identity domain for the default security realm in the domain. The purpose of the administrative identity domain is to distinguish between global domain users and partition users. WebLogic Server system administrators belong to this administrative identity domain.

The administrative identity domain is shown on the Oracle WebLogic Server Administration Console **Domain > Security** page.

- **Primary identity domain:** The primary identity domain configured for a partition. The primary identity domain is used as the default identity domain when authenticating partition users and for determining ownership of partition resources.

This primary identity domain is shown in the WebLogic Server Administration Console, on the Authentication providers provider-specific pages.

The user store's representation of an identity domain can be anything supported by the underlying technology. For example:

- You can create a separate Lightweight Directory Access Protocol (LDAP) instance for each identity domain.
- An identity domain field can be added to user records in a database.
- Identity domains are represented as distinct sub-trees in the Users and Groups hierarchies of a single Oracle Internet Directory (OID) instance.

To enable partition administrators access to realm security data in a partition, the system administrator must configure a management identity domain on the realm. The

management identity domain can be the primary identity domain, the administrative identity domain, or another identity domain supported by an authenticator in the realm. A partition administrator with the identity domain that matches this management identity domain can access a subset of the security MBeans and can invoke management operations on the MBeans. Note that although the administrative identity domain can be the management identity domain, it is not necessary because the system administrator already has permission to manage the security realm.

Default Identity Domain Values

WebLogic Server MT creates a default administrative identity domain and a default primary identity domain as follows:

- **Administrative identity domain:** WebLogic Server MT creates a default identity domain named `idd_DOMAIN` in the default security realm. The administrative identity domain is created when you create the first partition in a domain in production mode.

If you create the first partition in a domain in development mode, then the default value is `null`. (This is the only case in which `null` is considered a valid identity domain.)
- **Primary identity domain:** WebLogic Server MT creates the default identity domain for a partition with the name of the partition prefixed with `idd` when you create the partition.

This primary identity domain is shown on the provider-specific WebLogic Server Administration Console page for the WebLogic Default Authenticator provider.

The default primary identity domain is a convenience feature for the Default Authenticator with the embedded LDAP. It is intended for testing and internal development use, not for a production environment. The embedded LDAP does not actually use identity domains.

Administrative Roles for Configuration and Management

You can use Fusion Middleware Control and WLST to connect directly to a domain partition instead of logging in at the domain level, as described in [Connecting Directly to Domain Partitions: Main Steps and WLST Example](#).

The configuration and management capabilities of the WebLogic Server administrative roles in a partitioned environment are shown in [Table 5-1](#).

Table 5-1 Administrative Roles for Configuration and Management

Role	Logged In to Domain	Logged In to Partition
Administrator	<p>Full control over domain resources, including partition configuration and management.</p> <p>Configure the management identity domain attribute on the Realm MBean.</p>	<p>Write access to partition-owned MBeans.</p> <p>Read-only access to own <code>PartitionMBean.Realm</code> and <code>PartitionMBean.PrimaryIdentityDomain</code> attributes.</p> <p>Read-only access to own <code>RealmMBean</code> and its children.</p> <p>The partition administrator in the management identity domain:</p> <ul style="list-style-type: none"> • Can invoke methods on a subset of the security and realm Runtime MBeans in its own partition. • Can invoke methods on a subset of management operations on the security provider MBeans in its own partition.
Deployer	Configure resources within the domain and partitions, deploy/redeploy/undeploy/start/stop applications within the domain and partitions.	Configure resources within the partition, deploy/redeploy/undeploy/start/stop applications within the partition.
Operator	Start/stop servers, partitions, resource groups.	Start/stop partition and resource group.
Monitor	Read-only access to domain and partition resources.	Read-only access to partition resources.

Configuring the Administrative Identity Domain: Main Steps and WLST Example

If you are using the default security realm, and only the Default Authenticator is used to store WebLogic Server system administrator credentials, then you can accept the default `idd_DOMAIN` value. However, if your chosen user store for the default realm supports identity domains, and you have configured an identity domain for WebLogic Server administrators, then you need to configure at least one authentication provider in the default realm to match your administrative identity domain value.

The administrative identity domain is shown on the WebLogic Server Administration Console **Domain > Security** page.

From the WebLogic Server Administration Console:

1. Select the Authentication provider in the default realm and specify an existing administrative identity domain.
 - a. In the navigation pane, select **Security Realms**.
 - b. Select the default security realm.

- c. Select the **Providers** page.
 - d. Select the Authentication provider.
 - e. Select the **Provider Specific** page.
 - f. Enter the name of a valid administrative identity domain in the **Identity Domain** field.
 - g. Save your changes.
2. Go to the **Domain > Security** page and change the **Administrative Identity Domain:** value to the name of the administrative identity domain.
 3. Save your changes.
 4. Restart the Administration Server.

Configuring the Administrative Identity Domain: WLST Example

The example does the following:

1. Gets the domain MBean.
2. Gets the security configuration for the domain.
3. Sets the administrative identity domain for the security configuration.
4. Gets the default realm for the security configuration.
5. Looks up the default Authentication provider for the realm.
6. Sets the identity domain for the default Authentication provider.

Note:

You must restart the Administration Server. Setting the administrative identity domain value is a nondynamic change and requires a server restart.

```
edit()
startEdit()
wls:/base_domain/edit/ !> domain=getMBean('/')

wls:/base_domain/edit/ !> ADMIN_IDD = "Admin-idd"
wls:/base_domain/edit/ !> secure=domain.getSecurityConfiguration()
wls:/base_domain/edit/ !> secure.setAdministrativeIdentityDomain(ADMIN_IDD)
wls:/base_domain/edit/ !> realm = secure.getDefaultRealm()
wls:/base_domain/edit/ !> defAtn =
realm.lookupAuthenticationProvider('DefaultAuthenticator')
wls:/base_domain/edit/ !> defAtn.setIdentityDomain(ADMIN_IDD)
```

Configuring Security Realms and Primary Identity Domains: Main Steps and Examples

You are not required to create a security realm. Partitions can share a security realm, or use the default realm. However, the best practice is to create a security realm for

each partition. This is particularly true if you want to use identity domains to achieve isolation between partitions.

If your user store does support identity domains, then configure the primary identity domain and the management identity domain as described in this section.

Perform the following steps to configure security realms and identity domains in a domain partition. Note that some of the steps require the WebLogic Server Administration Console.

1. Create a security realm. The name must be unique within the domain.

If you are using Fusion Middleware Control, then go to **WebLogic Domain > Security > Security Realms**. The default providers are created on your behalf in the new realm.

If you are using the WebLogic Server Administration Console, then in the navigation pane, select **Security Realms**. You will probably find it most convenient to select the option to create the default providers in the new realm.

2. Add partition-specific users to the security realm.

If you are using Fusion Middleware Control, go to **WebLogic Domain > Security > Users and Groups**. Select the security realm that you just created and on the **Users** and **Groups** pages, click **Create** to add users and groups to the realm.

If you are using the WebLogic Server Administration Console, then in the navigation pane, select **Security Realms**. Select the security realm that you just created, and then the **Users and Groups** page. On the **Users** and **Groups** pages, click **New** to add users and groups to the realm.

3. Configure your providers in the security realm, as described in Manage Security Providers in the *Oracle WebLogic Server Administration Console Online Help*.

Perform this step from the WebLogic Server Administration Console.

4. Optionally, select the Authentication provider that you just created and specify an existing identity domain. Identity domains are an optional feature that you can use only if your user store supports them.

Perform this step from the WebLogic Server Administration Console.

- a. Select the **Provider Specific** page.
 - b. In the **Identity Domain** field, enter the name of a valid identity domain.
 - c. Save your changes.
5. When you later create a partition as described in [Configuring Domain Partitions](#), or if you created a partition when you created the domain using the Domain Configuration Wizard.

- a. Specify the security realm that you created.

If you use the WebLogic Server Administration Console to configure a partition that you created using the Domain Configuration Wizard, select the security realm in the **User Realm** field on the **Partition > Configuration > General** page.

- b. Specify the identity domain that you set for the Authentication provider.

If you use Fusion Middleware Control to create the domain partition, then specify the primary identity domain on the **Create Domain Partition: General** page. If you do not specify a value, then WebLogic Server MT creates the

default identity domain for a partition with the name of the partition prefixed with `idd` when you create the partition.

If you use the WebLogic Server Administration Console to create the domain partition, then WebLogic Server MT creates the default identity domain with the name of the partition prefixed with `idd`. After you create the partition, and before you activate your changes, go to the **Partition >Configuration >General** page and change the default name in the **Primary Identity Domain** field to your actual identity domain name.

- c. Set the management identity domain attribute in the realm to the same value that you used for the primary identity domain for the partition.

Perform this step from the WebLogic Server Administration Console.

Select **Security Realms** in the navigation pane. Select the security realm for the partition, then select **Configuration >General** and click **Advanced**. In the **Management Identity Domain** field, enter the name of the primary identity domain configured for the partition, and click **Save**.

Configuring Security Realms and Primary Identity Domains: WLST Example

The example does the following:

1. Gets the domain MBean.
2. Gets the security configuration for the domain.
3. Sets the administrative identity domain for the security configuration.
4. Gets the default realm for the security configuration.
5. Looks up the default Authentication provider for the realm.
6. Sets the identity domain for the default Authentication provider.
7. Creates a new realm for the partition.
8. Creates the required security providers for the new realm.
9. Creates a domain partition.
10. Creates a virtual target.
11. Sets the host name and URI prefix for the virtual target.
12. Adds the virtual target as an available target in the partition.
13. Sets the security realm for the partition.
14. Sets the primary identity domain for the partition.
15. Sets the management identity domain for the partition.
16. Creates the resource group.
17. Adds the virtual target to the resource group.
18. Activates the changes.
19. Starts the partition.

 **Note:**

You must restart the Administration Server before you can start the partition. Setting the administrative identity domain value is a nondynamic change and requires a server restart.

```

edit()
startEdit()
wls:/base_domain/edit/ !> domain=getMBean('/')

wls:/base_domain/edit/ !> ADMIN_IDD = "Admin-idd"
wls:/base_domain/edit/ !> secure=domain.getSecurityConfiguration()
wls:/base_domain/edit/ !> secure.setAdministrativeIdentityDomain(ADMIN_IDD)
wls:/base_domain/edit/ !> realm = secure.getDefaultRealm()
wls:/base_domain/edit/ !> defAtn =
realm.lookupAuthenticationProvider('DefaultAuthenticator')
wls:/base_domain/edit/ !> defAtn.setIdentityDomain(ADMIN_IDD)

wls:/base_domain/edit/ !> partrealm=secure.createRealm('partrealm')

wls:/base_domain/edit/ !>
partrealm.createAuthenticationProvider("DefaultAuthenticator", "weblogic.security.prov
iders.authentication.DefaultAuthenticator")
wls:/base_domain/edit/ !> defAtnP =
partrealm.lookupAuthenticationProvider('DefaultAuthenticator')
wls:/base_domain/edit/ !> defAtnP.setIdentityDomain('partID')
wls:/base_domain/edit/ !>
partrealm.createAuthenticationProvider("DefaultIdentityAsserter", "weblogic.security.p
roviders.authentication.DefaultIdentityAsserter")
wls:/base_domain/edit/ !>
partrealm.createAuthorizer("XACMLAuthorizer", "weblogic.security.providers.xacml.autho
rization.XACMLAuthorizer")
wls:/base_domain/edit/ !>
partrealm.createRoleMapper("XACMLRoleMapper", "weblogic.security.providers.xacml.autho
rization.XACMLRoleMapper")
wls:/base_domain/edit/ !>
partrealm.createAdjudicator("DefaultAdjudicator", "weblogic.security.providers.authori
zation.DefaultAdjudicator")
wls:/base_domain/edit/ !>
partrealm.createCredentialMapper("DefaultCredentialMapper", "weblogic.security.provide
rs.credentials.DefaultCredentialMapper")
wls:/base_domain/edit/ !> cert =
partrealm.createCertPathProvider("WebLogicCertPathProvider", "weblogic.security.provid
ers.pk.WebLogicCertPathProvider")
wls:/base_domain/edit/ !> partrealm.setCertPathBuilder(cert)
wls:/base_domain/edit/ !> partrealm.setManagementIdentityDomain('partID')
wls:/base_domain/edit/ !> pv =
partrealm.createPasswordValidator('SystemPasswordValidator', 'com.bea.security.provide
rs.authentication.passwordvalidator.SystemPasswordValidator')
wls:/base_domain/edit/ !> pv.setMinPasswordLength(8)
wls:/base_domain/edit/ !> pv.setMinNumericOrSpecialCharacters(1)

wls:/base_domain/edit/ !> peppart=domain.createPartition('Pep')
wls:/base_domain/edit/ !> vt=domain.createVirtualTarget('TestVT')
wls:/base_domain/edit/ !> vt.setHostNames(jarray.array([String('somehost')],String))

```



```
wls:/base_domain/edit/ !> vt.setUriPrefix('/foo')
wls:/base_domain/edit/ !> peppart.addAvailableTarget(vt)
wls:/base_domain/edit/ !> peppart.setRealm(partrealm)
wls:/base_domain/edit/ !> peppart.setPrimaryIdentityDomain('partID')
wls:/base_domain/edit/ !> peprg=peppart.createResourceGroup('TestRG')
wls:/base_domain/edit/ !> peprg.addTarget(vt)
wls:/base_domain/edit/ !> activate()
wls:/base_domain/edit/ !> startPartitionWait(peppart)
```

Configuring Security Realms and Primary Identity Domains: REST Example

For an example of creating domain partitions from Representational State Transfer (REST), see *Creating Partitions in Administering Oracle WebLogic Server with RESTful Management Services*.

This example demonstrates how to create a partition, including:

- A new security realm for the partition, including security providers, the primary identity domain, and the management identity domain.
- A virtual target for the cluster, on which the applications will run.
- A virtual target for the Administration Server, so that the Deployer role can create system resources and deploy applications.
- A resource group for each virtual target.
- Partition users in the Administrator, Deployer, Monitor, and Operator roles.

For an example that demonstrates how to set the management identity domain on a partition, see [Managing Users and Groups as a Partition Administrator: REST Example](#)

Setting the Identity Domain-Aware Providers Required Control: Main Steps

The WebLogic Server role mapping, authorization, credential mapping, and audit security providers must be identity-domain-aware to function correctly in an environment where identity domains are configured.

For example, an authorization provider that does not understand identity domains cannot correctly distinguish between two users with the same name but different identity domains, and therefore cannot make valid authorization decisions.

The standard WebLogic Server security providers are identity-domain-aware. However, deprecated providers, such as Default Role Mapper and Default Authorizer, are not identity-domain-aware.

If you have configured identity domains in your user store, you can force the use of identity-domain-aware providers by setting the **Identity Domain Aware Providers Required** control. Setting this control specifies that only role mapping, authorization, credential mapping, and audit providers that support identity domains are used, even if no identity domains are configured.

From the WebLogic Server Administration Console:

1. Go to the **Domain > Security** page.

2. Select the **Identity Domain Aware Providers Required** check box.
3. Save your changes.

Connecting Directly to Domain Partitions: Main Steps and WLST Example

Instead of logging in to the domain level, you can connect directly to a domain partition using both Fusion Middleware Control and WLST. When you do this, the user name that you specify is validated against the security realm and the management identity domain for that partition.

To login to a partition in Fusion Middleware Control:

1. Enter the URL in your Web browser. For example:
`http://host1.example.com:7001/em`
2. Enter the partition user name and password and click **Login to Partition**.
3. Enter the name of the partition in the **Partition Name** field and click **Sign in**.

See [Table 5-1](#) for the configuration and management capabilities of the WebLogic Server administrative roles when logged directly in to a partition.

Connecting Directly to Domain Partitions: WLST Example

You can use WLST to connect directly to a domain partition instead of logging in to the domain level. When you do this:

- The user name you specify is validated against the security realm for that partition.
See [Table 6-1](#) for the configuration and management capabilities of the WebLogic administrative roles when logged directly in to a partition.
- The initial view of the MBean hierarchy is partition-specific.

The WLST syntax is as follows. Note that only t3 connections are supported.

The initial view of the MBean hierarchy is partition-specific.

Using the Partition Name

```
wls:/offline> connect('username','password',"t3://system:port/partitions/partition-name")
```

For example:

```
wls:/offline> connect('weblogic','password',"t3://somehost:7001/partitions/Partition-0")
Connecting to t3://somehost:7001/partitions/Partition-0 with userid
weblogic ...
Successfully connected to partition "Partition-0".
wls:/base_domain/serverConfig/Partitions/Partition-0>
```

Using a Virtual Target

Specify a URL that matches a configured virtual target for the partition. For example, if you have a virtual target without a host name and a `uriPrefix` of `/foo`, then you can connect as follows:

```
connect('weblogic','password','t3://somehost:7001/foo')
```

MBean Hierarchy Is Partition-Specific

The MBean hierarchy after connecting is partition-specific:

```
wls:/base_domain/serverConfig/Partitions/Partition-0> ls()
dr-- AvailableTargets
dr-- CoherencePartitionCacheConfigs
dr-- DataSourceForJobScheduler
dr-- DataSourcePartition
dr-- DefaultTargets
dr-- JDBCSystemResourceOverrides
dr-- JMSSystemResourceOverrides
dr-- JTAPartition
dr-- MailSessionOverrides
dr-- ManagedExecutorServiceTemplates
dr-- ManagedScheduledExecutorServiceTemplates
dr-- ManagedThreadFactoryTemplates
dr-- PartitionLog
dr-- PartitionWorkManager
dr-- PartitionWorkManagerRef
dr-- ResourceGroups
dr-- ResourceManager
dr-- ResourceManagerRef
dr-- SelfTuning
dr-- SystemFileSystem
dr-- WebService

-r-- BatchJobsExecutorServiceName      null
-r-- DataSourceForJobScheduler          null
-r-- GracefulShutdownTimeout           0
-r-- IgnoreSessionsDuringShutdown       false
-r-- JobSchedulerTableName              WEBLOGIC_TIMERS
-r-- MaxConcurrentLongRunningRequests   50
-r-- MaxConcurrentNewThreads            50
-r-- Name                               Partition-0
-r-- ParallelDeployApplicationModules   false
-r-- ParallelDeployApplications         true
-r-- PartitionID                        085b48c2-6d70-434e-8200-f4eb
f28ca3a0
-r-- PartitionLifecycleTimeoutVal       120
-r-- PartitionWorkManager               null
-r-- PartitionWorkManagerRef            null
-r-- PrimaryIdentityDomain               idd_Partition-0
-r-- RCMHistoricalDataBufferLimit       250
-r-- ResourceDeploymentPlanPath          null
-r-- ResourceManager                    null
-r-- ResourceManagerRef                 null
-r-- StartupTimeout                      0
-r-- Type                                Partition
-r-- UploadDirectoryName                 C:\Oracle\Middleware\Oracle_
Home\user_projects\domains\base_domain\partitions\Partition-0\system\servers\Adm
inServer\upload\
```

Managing Security Data as a Partition Administrator: Main Steps and Examples

When logged in or connected to a partition as a member of the management identity domain, the partition administrator can manage the security realm data associated with the partition, such as managing users and groups, credential maps, roles, and policies. A partition administrator cannot change the configuration of the realm.

Note:

To enable security management capabilities for the partition administrator, the system administrator must set the management identity domain attribute on the realm to be the same as the primary identity domain. See [Configuring Security Realms and Primary Identity Domains: Main Steps and Examples](#).

In Fusion Middleware Control, partition administrators can manage users and groups. To invoke additional security management operations on the supported MBeans, they can use WLST, REST APIs, or JMX.

An overview of the process is as follows:

1. The system administrator configures a management identity domain on the realm. A partition administrator with the identity domain that matches this management IDD can invoke on the Security Runtime MBeans operations that control realm behavior. For example:
 - The system administrator configures the administrative identity domain on the Security Configuration MBean with the value `idd-DOMAIN`.
 - The system administrator configures partition `Partition1` in the domain. The primary identity domain configured on the Partition MBean has the value `idd-Partition1`. Users that authenticate to `Partition1` are part of the `idd-Partition1` identity domain.
 - The system administrator configures a `Partition1` realm in the domain and configures the management identity domain on the Realm MBean using the value `idd-Partition1`.
 - The system administrator creates the partition administration user and assigns it to the Administrators group.
 - Users with identity domain `idd-DOMAIN` (such as system administrators) and users with identity domain `idd-Partition1` (such as partition administrators) can access the security management operations on the Realm MBean and security provider MBeans contained by the realm.
2. The partition administrator can manage users and groups using Fusion Middleware Control.
3. The partition administrator can navigate to the realm via the Partition MBean Realm attribute.
4. The partition administrator can navigate to the security providers from the Realm MBean provider-related attributes.

5. The partition administrator can invoke on a subset of the methods in the security provider MBeans. For a list of the MBean mixin interfaces for security providers available to the partition administrator, see Table 6-3 in Security MBeans in *Developing Custom Management Utilities Using JMX for Oracle WebLogic Server*.

In a shared realm scenario with multiple partitions referring to one realm, the system administrator configures a management identity domain that matches the primary identity domain of one of the partitions. In this case, by default, only the users from the partition that matches can access the security management operations. The system administrator can also configure a separate authenticator that supports the management identity domain specifically and, in this case, users from all partitions that authenticate and specify the management identity domain can access the security management operations.

Table 5-2 lists the security management packages on which the partition administrator can invoke MBean operations when connected to the partition.

Table 5-2 Security Packages Accessible by a Partition Administrator

Package	Description
weblogic.management.security	Provides interfaces and classes for the control and monitoring of security providers in a WebLogic security realm.
weblogic.management.security.audit	Provides interfaces and classes for the control and monitoring of Audit security providers in a WebLogic security realm.
weblogic.management.security.authentication	Provides interfaces and classes for the control and monitoring of Authentication security providers in a WebLogic security realm.
weblogic.management.security.authorization	Provides interfaces and classes for the control and monitoring of Authorization, Adjudication, and Role Mapping security providers in a WebLogic security realm.
weblogic.management.security.credentials	Provides interfaces and classes for the control and monitoring of Credential Mapping security providers in a WebLogic security realm.
weblogic.management.security.pk	Provides interfaces and classes for the control and monitoring of certification path security providers in a WebLogic security realm.
weblogic.management.utils	Provides interfaces and classes for the listing of data returned from security providers in a WebLogic security realm.

For details about how to manage the security realm using JMX, see Managing Security Realms with JMX in *Developing Custom Management Utilities Using JMX for Oracle WebLogic Server*.

Managing Users and Groups as a Partition Administrator: Fusion Middleware Control Example

To manage users and groups using Fusion Middleware Control:

1. Log into the partition as the partition administrator. Note that the partition administrator must be in the management identity domain set on the partition realm.
2. From the **Domain Partition** menu, select **Security**, then **Users and Groups**.

3. To create or monitor users, click the **Users** tab (if it is not already selected).
 - To create a new user, click **Create**. In the Create a User page, enter the user name, description, provider, and password, then click **Create**.
 - To edit the settings for an existing user, select the user name in the table. In the Settings for User page, edit the General Settings, Password, Attributes, and Groups as desired, then click **Save**.
4. To create or monitor groups, click the **Groups** tab.
 - To create a new group, click **Create**. In the Create a Group page, enter the name, description, and provider for the group and click **Create**.
 - To edit the settings for an existing group, select the group name in the table. In the Settings for Group page, edit the General Settings and Membership for the group as desired, then click **Save**.

Creating a New User as a Partition Administrator: WLST Example

The following example demonstrates how to log into the partition `Partition1` as the partition administrator, connect to the realm partition and authentication provider `Partition1Realm/auth_provider2`, list the available operations, and add a new user to the realm.

1. Connect to the partition as the partition administrator `P1Admin` as described in [Connecting Directly to Domain Partitions: Main Steps and WLST Example](#).

For example:

```
wls:/offline> connect('P1Admin','password','t3://somehost:7001/partitions/Partition1')
Connecting to t3://somehost:7001/partitions/Partition1 with userid P1Admin ...
Successfully connected to partition "Partition1".
```

2. Navigate to the authentication provider in the partition realm.

For example:

```
wls:/base_domain/serverConfig/Partitions/Partition1> cd ("Realm/Partition1Realm")
wls:/base_domain/serverConfig/Partitions/Partition1/Realm/Partition1Realm> cd("AuthenticationProviders/auth_provider2")
```

3. List the available operations on the MBeans.

For example:

```
wls:/base_domain/serverConfig/Partitions/Partition1/Realm/Partition1Realm/AuthenticationProviders/auth_provider2> ls('o')
```

-r-x	addMemberToGroup	Void :
	String(groupName),String(memberUserOrGroupName)	
-r-x	advance	Void : String(cursor)
-r-x	changeUserPassword	Void :
	String(userName),String(oldPassword),String(newPassword)	
-r-x	close	Void : String(cursor)
-r-x	createGroup	Void :
	String(groupName),String(description)	
-r-x	createUser	Void :
	String(userName),String(password),String(description)	
-r-x	exportData	Void :

```

String(format),String(filename),java.util.Properties
-r-x  getCurrentName                String : String(cursor)
-r-x  getGroupDescription            String : String(groupName)
-r-x  getSupportedUserAttributeType
javax.management.openmbean.OpenType : String(userAttributeName)

-r-x  getUserAttributeValue          Object :
String(userName),String(userAttributeName)
-r-x  getUserDescription             String : String(userName)
-r-x  groupExists                   Boolean : String(groupName)
-r-x  haveCurrent                   Boolean : String(cursor)
-r-x  importData                    Void :
String(format),String(filename),java.util.Properties
-r-x  isMember                       Boolean :
String(parentGroupName),String(memberUserOrGroupName),Boolean(recursive)
-r-x  isUserAttributeNameSupported Boolean :
String(userAttributeName)
-r-x  listAllUsersInGroup            String[] :
String(groupName),String(userNameWildcard),Integer(maximumToReturn)
-r-x  listGroupMembers              String :
String(groupName),String(memberUserOrGroupNameWildcard),Integer(maximumToReturn)
-r-x  listGroups                    String :
String(groupNameWildcard),Integer(maximumToReturn)
-r-x  listMemberGroups              String :
String(memberUserOrGroupName)
-r-x  listUsers                     String :
String(userNameWildcard),Integer(maximumToReturn)
-r-x  removeGroup                   Void : String(groupName)
-r-x  removeMemberFromGroup         Void :
String(groupName),String(memberUserOrGroupName)
-r-x  removeUser                   Void : String(userName)
-r-x  resetUserPassword             Void :
String(userName),String(newPassword)
-r-x  setGroupDescription            Void :
String(groupName),String(description)
-r-x  setUserAttributeValue         Void :
String(userName),String(userAttributeName),Object(newValue)
-r-x  setUserDescription             Void :
String(userName),String(description)
-r-x  userExists                    Boolean : String(userName)

```

4. Add a new user in the realm.

For example:

```

wls:/base_domain/serverConfig/Partitions/Partition1/Realm/Partition1Realm/
AuthenticationProviders/auth_provider2>
cmo.createUser("myuser2","mypassword2","A user in the Partition1 default
authenticator")

```

5. Verify that the new user was created successfully.

For example:

```

wls:/base_domain/serverConfig/Partitions/Partition1/Realm/Partition1Realm/
AuthenticationProviders/auth_provider2>
if cmo.userExists('myuser2'):
  print "User myuser2 added"
...
User myuser2 added

```

Managing Users and Groups as a Partition Administrator: REST Example

For an example of managing security data as a partition administrator in a partition from Representational State Transfer (REST), see *Configuring Users in Administering Oracle WebLogic Server with RESTful Management Services*.

This example demonstrates how the partition administrator creates deployer, operator, and monitor users for the partition and assigns them to their respective groups.

Configuring SSL in Domain Partitions

Secure Sockets Layer (SSL) configuration is not partition-specific. All partitions use the standard WebLogic Server SSL configuration. See *Configuring SSL in Administering Security for Oracle WebLogic Server*.

However, note the following important differences:

- If you configure an explicit or offset port for the virtual target as described in [Configuring Virtual Targets](#), then SSL is not supported for that virtual target. This means that clients (for example, Remote Method Invocation (RMI) clients) cannot communicate directly over SSL to a partition that uses port-based routing.
- CertPath providers configured in nondefault security realms, such as security realms associated with one or more partitions, are not run as part of the extra SSL validation. Only the default security realm's CertPath provider is run.
- SSL is not supported in multitenant environments for older WebLogic Server clients. WebLogic Server clients prior to 12.2.x (that is, 10.3.6 and 12.1.x), are not multitenant aware. Therefore, to connect an older client to a multitenant partition, you must define an explicit port number on the virtual target for the partition as described in [Using Partition Channels with Virtual Targets](#). SSL is not supported for explicit or offset ports on virtual targets.

Configuring Security in Domain Partitions: Related Tasks and Links

- Manage Security Providers in the *Oracle WebLogic Server Administration Console Online Help*
- Create Users in the *Oracle WebLogic Server Administration Console Online Help*
- Configuring SSL in *Administering Security for Oracle WebLogic Server*

6

Configuring Oracle Traffic Director

In a typical deployment scenario, Oracle Traffic Director distributes incoming client requests to Oracle WebLogic Server. In a WebLogic Server MT environment, Oracle Traffic Director distributes incoming client requests to WebLogic Server MT partitions by coordinating its configuration with WebLogic Server MT partition management, automatically and without any explicit user action.

Note:

WebLogic Server Multitenant domain partitions, resource groups, resource group templates, virtual targets, and Resource Consumption Management are deprecated in WebLogic Server 12.2.1.4.0 and will be removed in the next release.

This chapter includes the following sections:

- [Configuring Oracle Traffic Director: Overview](#)
- [Configuring Oracle Traffic Director: Main Steps](#)
- [Oracle Traffic Director: Troubleshooting](#)
- [Configuring Oracle Traffic Director: Related Information](#)

Configuring Oracle Traffic Director: Overview

To employ Oracle Traffic Director multitenant support, you must perform an initial, one-time Oracle Traffic Director configuration.

Consider the following deployment topologies:

- Oracle WebLogic Server MT and Oracle Traffic Director in separate domains
In this topology, Oracle Traffic Director resides in a separate domain from the WebLogic Server MT domains. These domains can be on different hosts. The Oracle Traffic Director instance that exists in its domain distributes the client requests to multiple WebLogic Server MT domains that exist on different hosts. Even though Oracle Traffic Director is in a separate domain, it must be collocated with WebLogic Server for its management. See *Setting up an Administration Domain in Administering Oracle Traffic Director*.
- Oracle WebLogic Server MT and Oracle Traffic Director in a single domain
In this topology, Oracle Traffic Director is in the same domain as the WebLogic Server MT domain. The Oracle Traffic Director instance exists in the same WebLogic Server MT domain and distributes the client requests to it. In this topology also, Oracle Traffic Director must be collocated with WebLogic Server MT for its management. See *Setting up an Administration Domain in Administering Oracle Traffic Director*.

In summary, you must:

- Install Oracle Traffic Director, collocated with WebLogic Server MT, and create an Oracle Traffic Director domain. See [Creating the Domain for Oracle Traffic Director](#).
- Perform a one-time, initial Oracle Traffic Director configuration to enable Oracle Traffic Director multitenancy features. This includes creating an Oracle Traffic Director MT configuration and instance, and registering the Oracle Traffic Director runtime with the lifecycle manager (LCM). See [Creating an Oracle Traffic Director MT Configuration and Instance](#) and [Registering the Oracle Traffic Director Runtime](#).
- Use the registered Oracle Traffic Director runtime when creating a load balancer configuration for WebLogic Server MT partitions during partition creation. Upon completion, the LCM coordinates the orchestration with Oracle Traffic Director appropriately, as described in [Lifecycle Management](#).

Oracle Traffic Director Partitions

When you create a WebLogic Server MT partition using Fusion Middleware Control, a corresponding Oracle Traffic Director partition is created for you. The Oracle Traffic Director partition is simply a grouping with the same name as the partition and the resource group. Fusion Middleware Control provides a summary table with the list of Oracle Traffic Director partitions to identify the Oracle Traffic Director artifacts that are mapped to partitions and resource groups. You can also list the Oracle Traffic Director partitions using WLST. See `otd_listPartitions` and `otd_listResourceGroups` in *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

Oracle Traffic Director artifacts map to WebLogic Server MT artifacts as follows:

- Each cluster maps to an origin-server pool.
- The host names of a virtual target that is associated with the partitions and/or resource groups map to a virtual server.
- The URI-prefix of the virtual target maps to a route within the virtual server corresponding to the host name of the virtual target.

For descriptions of Oracle Traffic Director artifacts, see Oracle Traffic Director Terminology in *Administering Oracle Traffic Director*.

Monitoring

Metrics are gathered for each partition. A system administrator can access the partition metrics using either Fusion Middleware Control or WLST. See *Methods for Monitoring Oracle Traffic Director Instances* in *Administering Oracle Traffic Director*.

Logging

Oracle Traffic Director has a separate access log for each partition. The access log file name for the partition is the same as the partition name itself.

You can view and manage logs using Fusion Middleware Control and WLST. See *Viewing Logs Using Fusion Middleware Control* and *Viewing Logs Using WLST* in *Administering Oracle Traffic Director*.

Configuring Oracle Traffic Director: Main Steps

To configure Oracle Traffic Director in a WebLogic Server MT environment, you must create an Oracle Traffic Director domain, an Oracle Traffic Director MT configuration and instance, and then register the Oracle Traffic Director runtime with the lifecycle manager (LCM).

- [Creating the Domain for Oracle Traffic Director](#)
- [Creating an Oracle Traffic Director MT Configuration and Instance](#)
- [Registering the Oracle Traffic Director Runtime](#)

Creating the Domain for Oracle Traffic Director

To create an Oracle Traffic Director MT configuration and instance, you must first create a WebLogic Server MT domain and extend it for Oracle Traffic Director using the restricted Oracle Java Required Files (JRF) template. Then, using either WLST or Fusion Middleware Control, you can create Oracle Traffic Director configurations and instances.

Create an Oracle WebLogic Server MT domain as follows:

- In large enterprise deployments, where a single Oracle Traffic Director instance distributes client requests to multiple Oracle WebLogic Server MT domains, you will want to create separate domains for Oracle WebLogic Server MT and Oracle Traffic Director.

For example, using two machines (m_1 and m_2), if you want to have an Oracle Traffic Director domain on m_1 and WebLogic Server MT domain on m_2 :

Create the Oracle Traffic Director domain on m_1 as follows:

1. Install WebLogic Server MT with JRF in `$ORACLE_HOME`.
2. Install Oracle Traffic Director in the same `$ORACLE_HOME`.
3. Call the Configuration Wizard.
4. Select the **Oracle Traffic Director - Restricted JRF** template for Oracle Traffic Director and proceed with the domain creation.

With these steps you create a WebLogic Server MT domain and extend it for Oracle Traffic Director, so that you can proceed with Oracle Traffic Director configurations and instances creation. Note that even in the Oracle Traffic Director domain, WebLogic Server MT with JRF must be installed and the WebLogic Server MT domain must be created and extended for Oracle Traffic Director.

To create the WebLogic Server MT domain on m_2 :

1. Install WebLogic Server MT in `$ORACLE_HOME` (there is no need for WebLogic Server MT with JRF).
 2. Call the Configuration Wizard.
 3. Create a basic WebLogic Server MT domain.
- In a collocated domain, you install Oracle Traffic Director into the same `ORACLE_HOME` where you have installed WebLogic Server MT.

For example, if you want to have both Oracle Traffic Director and WebLogic Server MT in a single domain on machine `m1`:

1. Install WebLogic Server MT with JRF in `$ORACLE_HOME`.
2. Install Oracle Traffic Director in the same `$ORACLE_HOME`.
3. Call the Configuration Wizard.
4. Select the **Oracle Traffic Director - Restricted JRF** template for Oracle Traffic Director and proceed with the domain creation.

In this deployment scenario, both WebLogic Server MT and Oracle Traffic Director are in the same domain and Oracle Traffic Director manages the WebLogic Server MT partitions that are created within this domain.

When using the Configuration Wizard to create the domain, you must select to create a new domain, and in the Templates dialog, you must select the **Oracle Traffic Director - 12.2.1 Restricted JRF** template.

For detailed steps to install and configuration the domain, see *Oracle Traffic Director Installation Guide*.

Creating an Oracle Traffic Director MT Configuration and Instance

After creating the Oracle Traffic Director domain (actually, a WebLogic Server MT domain that is extended for Oracle Traffic Director), you must create a bootstrap Oracle Traffic Director configuration using Fusion Middleware Control or WLST.

Using Fusion Middleware Control to Create the Configuration and Instance

For more information about creating an Oracle Traffic Director MT configuration and instance using Fusion Middleware Control, see *Creating a Configuration Using Fusion Middleware Control and Creating Oracle Traffic Director Instances Using Fusion Middleware Control* in *Administering Oracle Traffic Director*.

Using WLST to Create the Configuration and Instance

For more information about creating an Oracle Traffic Director MT configuration and instance using WLST, see *Creating a Configuration Using WLST and Creating an Oracle Traffic Director Instance Using WLST* in *Administering Oracle Traffic Director*.

Registering the Oracle Traffic Director Runtime

Using Fusion Middleware Control, register the Oracle Traffic Director runtime to enable the lifecycle events or operations.



Note:

The Oracle Traffic Director runtime must be registered in the WebLogic Server MT domain. Using Fusion Middleware Control, log in to the WebLogic Server MT domain before registering the Oracle Traffic Director runtime.

1. From the **WebLogic Domain** menu, select **Environment > OTD Runtimes**.

2. Click **Register Runtime**.

Specify a name for the new Oracle Traffic Director runtime and provide information (host, port and credentials) on where this runtime is located. You can specify any Oracle Traffic Director runtime name, but you must select the same runtime name when creating a load balancer configuration for the WebLogic Server MT partition during partition creation.

You also need to provide the name of an existing Oracle Traffic Director configuration that will be used for MT. For example, if you wanted to use the configuration that you created in [Using Fusion Middleware Control to Create the Configuration and Instance](#), then you must specify the same name that you specified during the Oracle Traffic Director configuration creation.

Note:

In the Register Runtime dialog, you must specify the Admin Server host and port details of the Oracle Traffic Director domain and the (Oracle Traffic Director domain) Administration Server credentials.

Oracle Traffic Director: Troubleshooting

WebLogic Server provides general debugging tips, frequently asked questions, and corrective actions you can take if the WebLogic Server MT and Oracle Traffic Director components become unsynchronized.

Before associating Oracle Traffic Director with a WebLogic Server MT partition, verify the following:

- The WebLogic Server MT Administration Server and Node Manager are up and running in an Oracle Traffic Director domain.
- The WebLogic Server MT Administration Server is able to reach the Node Manager without any issues.
- You have created an Oracle Traffic Director configuration and the corresponding instance to be used for MT. See [Creating an Oracle Traffic Director MT Configuration and Instance](#).
- You have registered the correct Oracle Traffic Director runtime with the lifecycle manager (LCM). See [Registering the Oracle Traffic Director Runtime](#).

Frequently Asked Questions

The following responses address frequently asked questions and issues. These FAQs are relevant to configuring Oracle Traffic Director for MT and partition management using LCM with Oracle Traffic Director.

- Can I use an existing Oracle Traffic Director configuration for MT?
Yes. However, the existing configuration name must be specified while registering the Oracle Traffic Director runtime with the LCM, using the runtime property called `configuration`. If it is not specified, then the name defaults to `mt`.
- How do I check whether Oracle Traffic Director is successfully associated with a WebLogic Server MT partition?

If the association is successful, then Oracle Traffic Director artifacts such as virtual server, route and such, are created for the WebLogic Server MT partition in Oracle Traffic Director.

Call `otd_listPartitions` and `otd_listResourceGroups` WLST commands in the Oracle Traffic Director domain to verify a successful association.

- Is it necessary to create an Oracle Traffic Director partition explicitly if I use the low-level Representational State Transfer (REST) APIs?

Yes. Fusion Middleware Control implicitly creates the Oracle Traffic Director partition, but the low-level REST APIs do not.

The Oracle Traffic Director partition name must be the same as the WebLogic Server MT partition name.

Note that the Oracle Traffic Director partition is not a functional artifact. It is used only to logically group all the Oracle Traffic Director artifacts that serve requests to a WebLogic Server MT partition.

- How can I determine whether Oracle Traffic Director is notified by the LCM?

The Oracle Traffic Director plug-in will log debug information if it is notified by the LCM. A sample log message:

```
<[com.oracle.weblogic.lifecycle.plugin.otd.OTDUtil:log] OTDLifecyclePlugin :  
Associating OTD with the WLS MT partition>
```

- How do I enable debugging for the Oracle Traffic Director plug-in?

Set the WebLogic Server MT Administration Server domain log level to `Debug`.

- In the Oracle WebLogic Server Administration Console, select **Environment > Servers > Logging > Advanced** and set the required severity levels (Minimum severity to log, Log file severity level, and such) to `Debug`.
- Using WLST: `cd('/Servers/AdminServer/Log/AdminServer')` and set the required severity levels to `Debug` (`cmo.setLoggerSeverity('Debug')`).

- I have changed the host name and URI-prefix value of a virtual target that a WebLogic Server MT partition is targeted to but Oracle Traffic Director did not get updated. Why?

The host name and URI-prefix of a virtual target are nondynamic attributes which require a partition restart to be effective. Restart the WebLogic Server MT partition to update Oracle Traffic Director.

- I have added a new resource group to the existing WebLogic Server MT partition but Oracle Traffic Director did not get updated. Why?

This is a known issue. See Oracle Traffic Director Is Not Being Updated with Resource Group Changes in *Release Notes for Oracle WebLogic Server*.

- How can I synchronize WebLogic Server MT and Oracle Traffic Director if they become unsynchronized?

Call the `sync` LCM REST API.

```
curl -v \  
--user $WLS_DOMAIN_ADMIN_USERNAME:$WLS_DOMAIN_ADMIN_PASSWORD \  
-H X-Requested-By:MyClient \  
-H Accept:application/json \  
-H Content-Type:application/json \  
-X POST http://$WLS_DOMAIN_ADMIN_HOSTNAME:$WLS_DOMAIN_ADMIN_PORT/management/  
lifecycle/latest/environments/$ENVIRONMENT_NAME/sync
```

Note that you must replace the \$ tokens appropriately.

- What if the `sync` REST API does not synchronize WebLogic Server MT and Oracle Traffic Director?

Dissociate and then reassociate the WebLogic Server MT partition with Oracle Traffic Director.

Using the REST APIs:

To dissociate:

```
curl -v \
--user $WLS_DOMAIN_ADMIN_USERNAME:$WLS_DOMAIN_ADMIN_PASSWORD \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-H Content-Type:application/json \
-d '{ "partition1Name": "$WLSPartition_Name", "partition1RuntimeName" :
"$WLS_RUNTIME_NAME", "partition2Name": "$OTDPartition_Name",
"partition2RuntimeName": "$OTD_RUNTIME_NAME", "properties" :[]}' \
-X POST http://$WLS_DOMAIN_ADMIN_HOSTNAME:$WLS_DOMAIN_ADMIN_PORT/management/
lifecycle/latest/environments/$ENVIRONMENT_NAME/dissociatePartitions
```

To associate:

```
curl -v \
--user $WLS_DOMAIN_ADMIN_USERNAME:$WLS_DOMAIN_ADMIN_PASSWORD \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-H Content-Type:application/json \
-d '{ "partition1Name": "$WLSPartition_Name", "partition1RuntimeName" :
"$WLS_RUNTIME_NAME", "partition2Name": "$OTDPartition_Name",
"partition2RuntimeName": "$OTD_RUNTIME_NAME", "properties" :[]}' \
-X POST http://$WLS_DOMAIN_ADMIN_HOSTNAME:$WLS_DOMAIN_ADMIN_PORT/management/
lifecycle/latest/environments/$ENVIRONMENT_NAME/associatePartitions
```

In Fusion Middleware Control:

- Go to **Domain Partition > Administration > Load Balancer Configuration**.
- Deselect the check box **Use OTD for load balancing** to dissociate.
- Select the same check box to associate again.
- Is there a separate log file for each Oracle Traffic Director partition?

Yes. The partition log file name is the same as the partition name (for example, `<OTD_PARTITION_NAME>.log`), which is located at `<OTD_DOMAIN_HOME>/servers/<OTD_INSTANCE_NAME>/logs`.

You can use the following WLST commands:

```
otd_getPartitionAccessLogProperties and
otd_setPartitionAccessLogProperties.
```

Configuring Oracle Traffic Director: Related Information

Using the LCM orchestration, only the Oracle Traffic Director artifacts that are required to successfully distribute incoming client requests to WebLogic Server MT partitions are configured automatically. These artifacts include virtual servers, origin-server pools (including the origin servers) and routes. Apart from this, all other configurations, such as enabling Secure Sockets Layer (SSL) for Oracle Traffic Director, creating and managing failover groups in Oracle Traffic Director and such, must be done explicitly

using the Oracle Traffic Director administration interfaces. See *Administering Oracle Traffic Director*.

7

Configuring Domain Partitions

Oracle WebLogic Server Multitenant (MT) implements an administrative and runtime slice of a domain, called a **domain partition**. A domain partition is a portion of a WebLogic Server domain that is dedicated to running application instances and related resources. You can create, configure, and manage domain partitions using Oracle Enterprise Manager Fusion Middleware Control (FMWC), WebLogic Scripting Tool (WLST), or Representational State Transfer (REST).

Note:

WebLogic Server Multitenant domain partitions, resource groups, resource group templates, virtual targets, and Resource Consumption Management are deprecated in WebLogic Server 12.2.1.4.0 and will be removed in the next release.

This chapter includes the following sections:

- [Creating Domain Partitions](#)
- [Creating Domain Partitions: Main Steps and Examples](#)
- [Managing Domain Partitions: Main Steps and Examples](#)
- [Controlling Domain Partitions: Main Steps and Examples](#)
- [Configuring Domain Partitions: Related Tasks and Links](#)

Creating Domain Partitions

You can create any number of domain partitions within a domain.

Note:

Oracle generally recommends no more than 10 partitions within a domain for best performance. However, your particular server environment may support a much higher number of partitions.

Creating Domain Partitions: Prerequisites

Before you can create a domain partition, you must satisfy the following prerequisites:

1. If you have not already done so, then create the domain that you plan to use.
Use the Oracle Enterprise Manager - Restricted JRF template to create the domain. This template automatically includes several other necessary templates.

The best practice is to create a new domain. If you plan to upgrade an existing domain, then you must still create a new security realm as described in [Configuring Security](#).

 **Note:**

If you use Fusion Middleware Control or the Oracle WebLogic Server Administration Console, then there is nothing specific to WebLogic Server MT when creating a cluster.

However, if you use WLST to create Managed Servers (configured or dynamic), the required Oracle Java Required Files (JRF) template is not applied. When you subsequently use Fusion Middleware Control to monitor the domain, monitoring does not work for the servers without the JRF template.

Therefore, for the WLST use case:

- a. Use WLST to create the cluster or Managed Server.
- b. Use the `applyJRF` command described in *WLST Command Reference for Infrastructure Components* to apply the JRF template to the Managed Servers.

2. Set the deployment mode for lifecycle management. You can perform this step from the WebLogic Server Administration Console or WLST.

From the WebLogic Server Administration Console:

- a. In the navigation pane, select the domain.
- b. Select the **Configuration > General** page.
- c. Expand the **Advanced** control.
- d. Change the **Lifecycle Management Services Availability** control to `Local Admin Server`.
- e. This is a nondynamic change. Restart the Administration Server.

From WLST:

```
edit()
startEdit()
cd('/LifecycleManagerConfig/mydomain')
cmo.setDeploymentType('admin')
activate()
```

This is a nondynamic change. Restart the Administration Server.

3. If you have not already done so, then create the security realm for the partition. Each partition must have a security realm. See [Configuring Security](#) for the steps to follow.
4. If you have not already done so, create one or more virtual targets. See [Configuring Virtual Targets](#) for the steps to follow.
5. If you want to use a resource group template with this domain partition, then create the resource group template first. See [Configuring Resource Group Templates](#) for the steps to follow.

6. If you are using Oracle Traffic Director for load balancing, then register the Oracle Traffic Director runtime configuration. See [Registering the Oracle Traffic Director Runtime](#) for the steps to follow.

Oracle Traffic Director: WebLogic Server Plug-in Enabled Prerequisite

If you are using Oracle Traffic Director for load balancing, then you must set the **WebLogic Plug-in Enabled** control in the WebLogic Server Administration Console.

You can set this control at one of three levels. The levels have a hierarchy. Setting it at one level serves as the default for the level below; setting it at the level below overrides the setting at the higher level. The levels are:

- Domain level: setting it at the domain level sets it for each cluster and Managed Server within the domain.
- Cluster level: setting it at the cluster level applies it to all the Managed Servers that are a part of the cluster. This overrides the value at the domain level.
- Individual Managed Server level: setting it at the Managed Server level overrides the value set at the cluster or domain levels.

You may find it easiest to set the WebLogic Plug-in Enabled control at the domain level, so that any partitions for which you want to use Oracle Traffic Director are affected, regardless of which clusters or Managed Servers are involved. Or, if you know that the partition that you want to use with Oracle Traffic Director will be targeted to only a specific cluster or Managed Server, then you can choose to set the WebLogic Plug-in Enabled control at the cluster or Managed Server level.

Domain Level

Using the WebLogic Server Administration Console, to set the WebLogic Plug-in Enabled control at the domain level:

1. In the navigation pane, select the domain.
2. Select the **Configuration** page, then **Web Applications**.
3. Set the **WebLogic Plug-in Enabled** control.
4. Save your changes.

Cluster Level

Using the WebLogic Server Administration Console, to set the WebLogic Plug-in Enabled control at the cluster level:

1. In the navigation pane, expand **Environment**.
2. Select **Clusters**.
3. Select the cluster that you want to manage.
4. Select the **General** page, then click **Advanced**.
5. Set the **WebLogic Plug-in Enabled** control.
6. Save your changes.

Managed Server Level

Using the WebLogic Server Administration Console, to set the WebLogic Plug-in Enabled control at the Managed Server level:

1. In the navigation pane, expand **Environment**.
2. Select **Servers**.
3. Select the servers that you want to manage.
4. Select the **General** page, then click **Advanced**.
5. Set the **WebLogic Plug-in Enabled** control.
6. Save your changes.

Creating Domain Partitions: Main Steps and Examples

Creating domain partitions includes several tasks, such as specifying a security realm for the partition, selecting virtual targets, creating resource groups, and specifying targets for the resource groups.

The following information describes the main steps for creating domain partitions.

1. Enter the partition name. The partition name must be unique within the domain.
2. Select the security realm for this partition.
The security realm can be unique to this partition, or shared by multiple partitions.
3. Optionally, enter a name for the primary identity domain for the partition. Or, you can choose not to enter a name and accept the default.
4. If you are using Oracle Traffic Director for load balancing, select the Oracle Traffic Director runtime configuration. You must have previously registered the Oracle Traffic Director runtime.
5. Select one or more existing virtual targets to be available for this domain partition to use. Multiple partitions cannot use the same virtual target. You can use a virtual target with only one partition or at the global (domain) level.
6. Select one of the existing virtual targets to use as the default if a resource group in this partition does not explicitly identify one.
7. Create the resource group. You can create the resource group in two ways:
 - Create a new resource group. When you finish creating the partition you must then edit this resource group as needed.
Enter the resource group name. The resource group name must be unique within the partition.
 - Create the new resource group based on a resource group template. (For information about resource group templates, see [Configuring Resource Group Templates](#).) The configuration is copied from the template to the new resource group.
Enter the resource group name. The resource group name must be unique within the domain.
Select the resource group template to use.
8. Select the virtual targets that this resource group will use.

9. If this is the first partition that you have created in this domain, and the domain is running in production mode, then restart the WebLogic Administration Server.
This step is needed only for the first partition that you create for a domain, and only when the domain is running in production mode.
10. Start the partition. Partitions are created in a shutdown state and need to be started for the resources in them to be accessible.
11. If you did not create the resource group from a resource group template, then edit the resource group as needed.

Proceed to [Configuring Resource Groups](#) for the main steps to follow.

To create a domain partition using Fusion Middleware Control, see Create domain partitions in the online help.

Creating Domain Partitions: WLST Example

The following example creates a domain partition `PartitionMBean` from the `DomainMBean`, called `pep`. It does the following:

1. Creates a domain partition.
2. Creates a virtual target.
3. Sets the host name and URI prefix for the virtual target.
4. Adds the virtual target as an available target in the partition.
5. Creates the resource group.
6. Adds the virtual target to the resource group.
7. Activates the changes.
8. Starts the partition.

Note:

If this is the first partition created in production mode, then you must restart the Administration Server before you can start the partition.

```
# Create Pep partition and ResourceGroup
edit()
startEdit()
wls:/base_domain/edit/ !> domain=getMBean('/')
wls:/base_domain/edit/ !> peppart=domain.createPartition('Pep')
wls:/base_domain/edit/ !> vt=domain.createVirtualTarget('TestVT')
wls:/base_domain/edit/ !> vt.setHostNames(jarray.array([String('localhost')],String))
wls:/base_domain/edit/ !> vt.setUriPrefix('/foo')
wls:/base_domain/edit/ !> peppart.addAvailableTarget(vt)
wls:/base_domain/edit/ !> peprg=peppart.createResourceGroup('TestRG')
wls:/base_domain/edit/ !> peprg.addTarget(vt)
wls:/base_domain/edit/ !> activate()
wls:/base_domain/edit/ !> startPartitionWait(peppart)
```

Creating Domain Partitions: REST Example

For an example of creating domain partitions using REST, see *Creating Partitions in Administering Oracle WebLogic Server with RESTful Management Services*.

Managing Domain Partitions: Main Steps and Examples

There are management tasks that you perform on the domain partition itself, and not on the associated resource groups. The tasks for configuring resource groups or resource group overrides are described in *Configuring Resource Groups and Configuring Resource Overrides*, respectively.

1. Select the domain partition that you want to manage.
2. Monitor the partition's performance and use.
3. View and change the virtual targets available for this domain partition.
4. View and change the resource groups configured in this partition.
5. View and change any applications that are deployed to the partition. (Strictly speaking, you deploy an application to a resource group in a partition, not to the partition itself.)
6. If you are using Oracle Traffic Director for load balancing, then view and change the Oracle Traffic Director runtime configuration used with this partition.
7. View and change any JDBC and JMS modules partition override configurations.
8. View and change any Partition Work Manager and Resource Consumption Manager configured for this partition.
9. View and change the security realm and default target for this partition.
10. Optionally, use the `Notes` attribute to specify additional information about this partition.

To manage a domain partition using Fusion Middleware Control, see *Configure domain partitions* in the online help.

Managing Domain Partitions: WLST Example

The example does the following:

1. Creates a domain partition.
2. Creates a virtual target.
3. Sets the host name and URI prefix for the virtual target.
4. Adds the virtual target as an available target in the partition.
5. Creates the resource group.
6. Adds the virtual target to the resource group.
7. Activates the changes.
8. Starts the partition.

 **Note:**

If this is the first partition created in production mode, then you must restart the Administration Server before you can start the partition.

9. Deploys the application `MySimpleEjb` to the resource group.

```
# Create Pep partition and ResourceGroup
edit()
startEdit()
wls:/base_domain/edit/ !> domain=getMBean('/')
wls:/base_domain/edit/ !> peppart=domain.createPartition('Pep')
wls:/base_domain/edit/ !> vt=domain.createVirtualTarget('TestVT')
wls:/base_domain/edit/ !> vt.setHostNames(jarray.array([String('localhost')],String))
wls:/base_domain/edit/ !> vt.setUriPrefix('/foo')
wls:/base_domain/edit/ !> peppart.addAvailableTarget(vt)
wls:/base_domain/edit/ !> peprg=peppart.createResourceGroup('TestRG')
wls:/base_domain/edit/ !> peprg.addTarget(vt)
wls:/base_domain/edit/ !> activate()

wls:/base_domain/edit/ !> startPartitionWait(peppart)

wls:/base_domain/edit/ !> deploy(appName='MySimpleEjb', path='c:/webservices/
MySimpleEjb.jar', partition='Pep', resourceGroup='TestRG',
deploymentOrder=10,securityModel='DDOnly')
:
Completed the deployment of Application with status completed
Current Status of your Deployment:
Deployment command type: deploy
Deployment State : completed
Deployment Message : no message
```

Managing Domain Partitions: REST Example

See the following REST examples:

- For an example of creating partition-scoped data sources and a JMS system resource using REST, see *Creating Partition-Scoped System Resources in Administering Oracle WebLogic Server with RESTful Management Services*.
- For an example of deploying partition-scoped applications using REST, see *Deploying Partition-Scoped Applications in Administering Oracle WebLogic Server with RESTful Management Services*.

Controlling Domain Partitions: Main Steps and Examples

You can reconfigure global and partition-specific resources without restarting the associated servers or clusters. Changes to one partition do not affect other partitions: you can create, start, stop, and delete partitions independently.

1. Select the partition that you want to control. You do not need to select a partition to import a partition.
2. Start the partition. All of the resource groups—and all of the applications deployed to those resource groups—are started.

3. Shut down the partition. All of the resource groups—and all of the applications deployed to those resource groups—are shut down. Shutting down a resource group causes the applications and resources in that resource group to stop running and to be removed from memory.

Shutting down the partition is the runtime equivalent of undeploying the application or resource, except that the configuration for the application or resource is not removed from the configuration file as it would be in a true undeploy operation. The configuration delivered to a Managed Server is also not removed.
4. Suspend the partition. The suspend operation gracefully transitions the partition from the `RUNNING` to `ADMIN` states.
5. Resume the partition. The resume operation gracefully transitions the partition from the `ADMIN` to `RUNNING` states.
6. Import a partition. You can import a partition into another domain with only a minimal number of configuration changes being required. See [Exporting and Importing Partitions](#).
7. Export a partition. You can export a partition from one domain (the source domain) and import it into another domain (the target domain). See [Exporting and Importing Partitions](#).
8. Delete a partition. When you delete a partition, all of the resource groups in the partition are deleted and all of the applications deployed to the partition are undeployed.

 **Note:**

You must shut down a partition before you can delete it.

To control a domain partition using Fusion Middleware Control, see Control domain partitions in the online help.

Actions That Require a Partition Restart

The following actions require a partition restart:

- Any change to a nondynamic attribute on a system resource in a resource group. For example, the URL of the JDBC database connection.
- Any override that affects a nondynamic attribute on a system resource. For example, the connection URL of the foreign JMS server.
- Any nontargeting change to a virtual target that is in use by a running partition. For example, the URI prefix.
- Any change to a resource deployment plan.

Controlling Domain Partitions: WLST Example

For examples of importing and exporting partitions with WLST, see [Exporting and Importing Partitions](#).

The following example uses the WLST `startPartitionWait()` command to start a domain partition called `pep` from the root `DomainMBean`. For more information about `startPartitionWait()`, see *WLST Command Reference for WebLogic Server*.

The `startPartitionWait()` command requires the partition to start. You can obtain the `PartitionMBean` from `Domain.createPartition()` or `Domain.lookupPartition()`, for example:

```
edit()
startEdit()
domain=getMBean('/')
startPartitionWait(domain.lookupPartition("Pep"))
[MBeanServerInvocationHandler]com.bea:Name=_5_START,Type=PartitionLifeCycleTaskRuntime,PartitionLifeCycleRuntime=Pep
```

Full Control with PartitionLifeCycleRuntimeMBean

The `startPartitionWait()` command is a convenience command. For full control of a partition life cycle, including shutting down a partition, use the `PartitionLifeCycleRuntimeMBean`. See `PartitionLifeCycleRuntimeMBean` in *MBean Reference for Oracle WebLogic Server*.

The following example gracefully shuts down the partition `pep`:

```
domainRuntime()
cd('DomainPartitionRuntimes')
cd('pep')
cd('PartitionLifeCycleRuntime')
cd('pep')
cmo.shutdown()
[MBeanServerInvocationHandler]com.bea:Name=_3_SHUTDOWN,Type=PartitionLifeCycleTaskRuntime,DomainPartitionRuntime=pep,PartitionLifeCycleRuntime=pep
```

The following example starts the partition `pep`:

```
domainRuntime()
cd('DomainPartitionRuntimes')
cd('pep')
cd('PartitionLifeCycleRuntime')
cd('pep')
cmo.start()
[MBeanServerInvocationHandler]com.bea:Name=_4_START,Type=PartitionLifeCycleTaskRuntime,DomainPartitionRuntime=pep,PartitionLifeCycleRuntime=pep
```

The following example suspends the partition `pep`:

```
domainRuntime()
cd('DomainPartitionRuntimes')
cd('pep')
cd('PartitionLifeCycleRuntime')
cd('pep')
cmo.suspend()
[MBeanServerInvocationHandler]com.bea:Name=_5_SUSPEND,Type=PartitionLifeCycleTaskRuntime,DomainPartitionRuntime=pep,PartitionLifeCycleRuntime=pep
```

Controlling Domain Partitions: REST Example

For an example of controlling domain partitions using REST, see *Starting and Stopping Partitions in Administering Oracle WebLogic Server with RESTful Management Services*.

Configuring Domain Partitions: Related Tasks and Links

- [Oracle WebLogic Server Multitenant](#)
- [Configuring Virtual Targets](#)
- [Configuring Resource Group Templates](#)
- [Configuring Resource Groups](#)
- [Configuring Resource Overrides](#)
- [Configuring Resource Consumption Management](#)
- [Configuring Oracle Coherence](#)
- [Configuring Partition Work Managers](#)
- [Exporting and Importing Partitions](#)
- [Monitoring and Debugging Partitions](#)

8

Configuring Resource Groups

You perform most of the configuration required for a partition when you configure its resource groups or resource group overrides. The tasks include configuring JDBC system data sources, JMS servers and resources, foreign JNDI providers, and such.

Note:

WebLogic Server Multitenant domain partitions, resource groups, resource group templates, virtual targets, and Resource Consumption Management are deprecated in WebLogic Server 12.2.1.4.0 and will be removed in the next release.

This chapter includes the following sections:

- [Configuring Resource Groups: Overview](#)
- [Creating Resource Groups: Main Steps and Examples](#)
- [Configuring Resource Groups: Main Steps and Examples](#)
- [Deleting Resource Groups: Main Steps and WLST Example](#)
- [Controlling Resource Groups: Main Steps and WLST Example](#)
- [Migrating Resource Groups: Main Steps and WLST Example](#)
- [Managing Administrative Applications and Resources in Partition Resource Groups](#)
- [Configuring Resource Groups: Related Tasks and Links](#)

Configuring Resource Groups: Overview

Resource groups gather together applications and the resources they use into a distinct administrative unit within the domain. Typically, the resources in a given resource group are related in some way. For example, they make up a single application suite.

The resources and applications have all the information needed to start or connect to those resources, including credentials for connecting to a data source and targeting information for applications. Applications deployed to a resource group should be ready to be started.

- [What Is in a Resource Group?](#)
- [Resource Groups and Overrides](#)
- [Resource Groups in Global Scope and Partitions](#)
- [Targeting a Resource Group to More Than One Target](#)

What Is in a Resource Group?

Resource groups are based on the [ResourceGroupMBean](#) and can include the following resources:

- General (name, scope, whether or not based on a resource group template)
- Deployments
- Services:
 - JDBC
 - Messaging
 - Mail sessions
 - Persistent stores
 - Foreign JNDI providers
 - OSGi frameworks
 - Diagnostics
- Targets
- Monitoring features (JDBC, messaging)
- Control features (JDBC, messaging, migration)
- Notes

Resource Groups and Overrides

Resource overriding lets you customize resources at the partition level. You can override resource settings that are derived from a resource group template.

There are two principal types of overrides:

- You can override resource settings for certain resources using resource override configuration MBeans and resource deployment plans.
- You can override the default application configuration for applications and modules defined to the resource group template by specifying a different deployment plan. The application or module is then redeployed using the new deployment plan for its application configuration.

For more information about resource overrides, see [Configuring Resource Overrides](#). For information about application overrides, see [Deploying Applications to Partition Resource Groups](#).

Resource Groups in Global Scope and Partitions

You can create resource groups at the domain level, or specific to a domain partition. If you create the resource group at the domain level, then it has a global scope, which is the equivalent of the domain level in a nonpartitioned environment. Applications or classes running at the domain level are available across the domain, but are not available in partitions.

If you create the resource group at the partition level, then it is scoped only to that partition. Applications or classes running at the partition level are available to the partition, but are not available at the domain level or in other partitions.

Targeting a Resource Group to More Than One Target

You can configure a resource group to have more than one virtual target. For example, you might target a resource group to virtual targets for `Cluster1`, `Cluster2`, and `Cluster3` so that the applications in the resource group run on all three clusters.

However, there are two notable restrictions:

- Some resource group configurations may have target-specific resources that do not apply across multiple targets. These resources include but are not limited to:
 - JMS server
 - Messaging bridge
 - Path service
 - JMS bridge destination
 - File store
 - JDBC store
 - JMS system resource

If you try to target a resource group to multiple virtual targets and any one of these resources is present, then WebLogic Server MT generates an error.

- You cannot target a resource group to more than one virtual target if the virtual targets target the same physical server.

For example, if resource group `RG` targets `VT1` and `VT2`, and both `VT1` and `VT2` target `Server1`, then WebLogic Server MT generates an error.

Creating Resource Groups: Main Steps and Examples

You can create a resource group in two ways: create a new resource group or create the resource group based on a resource group template.

Creating a new resource group creates the basic structure for the resource group. You must then edit this resource group as needed. Whereas, when creating a resource group from a resource group template, the configuration is copied from the template to the new resource group. You must then edit and override values from the resource group template as needed. For information about resource group templates, see [Configuring Resource Group Templates](#).

To create a new resource group at the partition or global scope:

1. Go to the partition or the domain level, as needed.

Note the following navigational differences:

- If you are using Fusion Middleware Control, you can go to **WebLogic Domain > Environment > Resource Groups** and create a resource group at either the partition level or domain level.

- If you are using the Oracle WebLogic Server Administration Console, go to **WebLogic Domain > Environment > Resource Groups** to create a resource group at the domain level.

Go to **WebLogic Domain > Domain Partitions > Partition > Resource Groups** to create a resource group at the partition level.

2. Create a new resource group.
3. You can define the following configuration settings for your new resource group:
 - Enter a name for the new resource group.
 - Select either partition level or domain level.
 - Optionally, select a resource group template to use for this new resource group.
 - Select the targets for this new resource group.
 - Enter notes for the new resource group.

These tasks are described in Create resource groups in *Administering Oracle WebLogic Server with Fusion Middleware Control*.

If you did not create the resource group from a resource group template, then the initial resource group configuration is a basic skeleton and you must configure it before you can use it. You perform most of the configuration required for a partition when you configure the resource group or resource group overrides. The tasks include configuring JDBC system data sources, JMS servers and resources, foreign JNDI providers, and so forth.

See [Configuring Resource Groups: Main Steps and Examples](#).

Creating Resource Groups: WLST Example

The example does the following:

1. Creates a domain partition.
2. Creates a virtual target.
3. Sets the host name and URI prefix for the virtual target.
4. Adds the virtual target as an available target in the partition.
5. Creates the resource group.
6. Adds the virtual target to the resource group.
7. Activates the changes.
8. Starts the partition.

Note:

If this is the first partition created in production mode, then you must restart the Administration Server before you can start the partition.

```
# Create Pep partition and ResourceGroup
edit()
startEdit()
```

```

wls:/base_domain/edit/ !> domain=getMBean('/')
wls:/base_domain/edit/ !> peppart=domain.createPartition('Pep')
wls:/base_domain/edit/ !> vt=domain.createVirtualTarget('TestVT')
wls:/base_domain/edit/ !> vt.setHostNames(jarray.array([String('localhost')],String))
wls:/base_domain/edit/ !> vt.setUriPrefix('/foo')
wls:/base_domain/edit/ !> peppart.addAvailableTarget(vt)
wls:/base_domain/edit/ !> peprg=peppart.createResourceGroup('TestRG')
wls:/base_domain/edit/ !> peprg.addTarget(vt)
wls:/base_domain/edit/ !> activate()
wls:/base_domain/edit/ !> startPartitionWait(peppart)

```

Creating Resource Groups: REST Example

For an example of creating resource groups using REST, see *Creating Partitions in Administering Oracle WebLogic Server with RESTful Management Services*.

In particular, see the section "Create a resource group for the new partition."

Configuring Resource Groups: Main Steps and Examples

Configuring resource groups may include several tasks, such as specifying general settings, deployment options, services, such as JDBC, JMS, mail sessions, persistent stores, and foreign JNDI providers, and specifying the targets for the resource groups.

- [Configuring Resource Group General Settings](#)
- [Configuring Resource Group Deployment Settings](#)
- [Configuring Resource Group Services Settings](#)
- [Configuring Resource Group Targets](#)
- [Configuring Resource Group Notes](#)

Configuring Resource Group General Settings

To view and define general resource group settings:

1. Select the resource group that you want to configure.
2. View and define general configuration settings for the resource group, such as:
 - Name
 - Scope
 - Resource Group Template
3. Save your changes.

These tasks are described in *Configure resource groups general settings in Administering Oracle WebLogic Server with Fusion Middleware Control*.

Configuring Resource Group Deployment Settings

To view and define resource group deployment settings:

1. Select the resource group that you want to configure.
2. You can take the following deployment actions:

- Deploy
 - Redeploy
 - Undeploy
 - Fetch deployment plan
 - Add override
 - Remove override
 - Start
 - Stop
3. Save your changes.

These tasks are described in Configure resource group deployment settings in *Administering Oracle WebLogic Server with Fusion Middleware Control*.

Deploying Applications to a Resource Group

Deploying an application makes its physical file or directory known to WebLogic Server.

To deploy an application to a resource group:

1. Select the resource group that you want to configure.
2. Locate the application that you want to deploy and choose whether to upload a deployment plan or create a new deployment plan.
3. Update the application attributes as desired. These attributes include:
 - Application name
 - Distribution
 - Source accessibility
4. Deploy the application.
5. Update the deployment settings or complete the deployment of this application.

These tasks are described in Deploy applications to a resource group in *Administering Oracle WebLogic Server with Fusion Middleware Control*.

Redeploying Applications to a Resource Group

Redeploying an application redeploys the archive file or exploded directory. Redeploy an application if you have made changes to it and want to make the changes available to WebLogic Server clients.

To redeploy an application or module to a resource group:

1. Select the resource group that you want to configure.
2. Select the application that you want to redeploy.
3. Decide whether to upload a deployment plan or to create a new deployment plan.
4. Update the application distribution as needed.
5. Optionally, edit the deployment plan to set more advanced deployment options, and save the deployment plan to your local disk.

6. Redeploy the application to complete redeployment of this application.

These tasks are described in *Redeploy applications to a resource group in Administering Oracle WebLogic Server with Fusion Middleware Control*.

Undeploying Applications from a Resource Group

Undeploying an application removes it from every target of the domain to which the application is deployed. After you undeploy an application from the domain, you must deploy it again if you want to make it available to WebLogic Server clients. To temporarily make applications unavailable to WebLogic Server clients, you can stop them instead of undeploying them.

To undeploy an application from a partition resource group:

1. Select the resource group that you want to configure.
2. Select the application that you want to undeploy from the deployed applications.
3. Undeploy the application.
4. If you later want to deploy the removed application, see [Deploying Applications to a Resource Group](#).

These tasks are described in *Undeploy applications from a resource group in Administering Oracle WebLogic Server with Fusion Middleware Control*.

Configuring Resource Group Services Settings

This section includes the following tasks:

- [Configuring Resource Group JDBC Settings](#)
- [Configuring Resource Group JMS settings](#)
- [Configuring Resource Group Mail Session Settings](#)
- [Configuring Resource Group Persistent Store Settings](#)
- [Configuring Resource Group Foreign JNDI Provider Settings](#)
- [Configuring Resource Group Diagnostic System Module Settings](#)

Configuring Resource Group JDBC Settings

To view configuration settings for the JDBC system resources that have been created in this resource group:

1. Select the resource group that you want to configure.
2. Go to **Services > JDBC**.

The read-only JDBC information for a resource group includes:

- Name
- JNDI name
- Type
- Targets
- Algorithm type

- Row prefetch enabled
 - Row prefetch size
 - Stream chunk size
3. Create or delete the system data sources as needed.
 4. Save your changes.

These tasks are described in Configure resource group JDBC Settings in *Administering Oracle WebLogic Server with Fusion Middleware Control*.

For information about configuring the JDBC system data sources that have been created in this resource group, see [Configuring JDBC](#).

Configuring Resource Group JMS settings

This section includes the following tasks:

- [Configuring JMS Server Settings](#)
- [Configuring SAF Agent Settings](#)
- [Configuring JMS Resource Settings](#)
- [Configuring JMS Module Settings](#)
- [Configuring Messaging Bridges](#)
- [Configuring JMS Bridge Destinations](#)
- [Configuring Path Services](#)

Configuring JMS Server Settings

To view configuration settings for the JMS servers that have been created for this resource group:

1. Select the resource group that you want to configure.
2. Go to **Services > Messaging > JMS Servers**.

The following read-only information is available for JMS servers already configured in this resource group:

- Name
 - Health
 - Health reason
 - Persistent store
 - Temporary template name
 - Bytes maximum
 - Messages maximum
 - Bytes threshold high
 - Bytes threshold low
 - Messages threshold high
3. Create or delete JMS servers as needed.

4. Save your changes.

These tasks are described in Configure JMS Server Settings in *Administering Oracle WebLogic Server with Fusion Middleware Control*.

For information about configuring the JMS servers that have been configured in this resource group, see [Configuring JMS Servers](#).

Configuring SAF Agent Settings

To view configuration settings for the Store-and-Forward (SAF) agents that have been created for this resource group:

1. Select the resource group that you want to configure.

2. Go to **Services > Messaging > SAF Agents**.

The following read-only information is available for the SAF agents configured in this resource group:

- Name
- Agent type
- Persistent store

3. Create or delete SAF agents as needed.

4. Save your changes.

These tasks are described in Configure SAF Agent Settings in *Administering Oracle WebLogic Server with Fusion Middleware Control*.

For information about configuring the SAF agents that have been configured in this resource group, see [Configuring Store-and-Forward Agents](#).

Configuring JMS Resource Settings

To monitor the resource settings for a resource group:

1. Select the resource group that you want to configure.

2. Go to **Services > Messaging > JMS Resources**.

The following read-only information is available for the JMS resources configured in this resource group:

- Name
- Type
- JMS module name
- JNDI name
- Subdeployment

3. Create or delete JMS resources as needed.

4. Save your changes.

These tasks are described in Configure SAF Agent Settings in *Administering Oracle WebLogic Server with Fusion Middleware Control*.

For information about configuring existing JMS resources, see [Configuring JMS System Resources and Application-Scoped JMS Modules](#).

Configuring JMS Module Settings

To configure the JMS modules for a resource group:

1. Select the resource group that you want to configure.
2. Go to **Services > Messaging > JMS Modules > JMS Modules**.

The following read-only information is available for the JMS modules configured in this resource group:

- Name
 - Queues
 - Topics
 - Connection factories
 - Distributed queues
 - Distributed topics
 - Foreign servers
 - Quotas
 - SAF error handlers
 - SAF imported destinations
 - SAF remote contexts
 - Templates
 - Uniform distributed queues
 - Uniform distributed topics
 - Destination keys
 - Type
3. Create or delete JMS modules as needed.
 4. Save your changes.

These tasks are described in Configure JMS module settings in *Administering Oracle WebLogic Server with Fusion Middleware Control*.

For information about configuring existing JMS modules, see [Configuring Messaging Components](#).

Configuring Messaging Bridges

To configure the messaging bridge settings for a resource group:

1. Select the resource group that you want to configure.
2. Go to **Services > Messaging > Messaging Bridges**.

The following read-only information is available for the messaging bridges configured in this resource group:

- Name
- Source bridge destination

- Target bridge destination
3. Create or delete messaging bridges as needed.
 4. Save your changes.

These tasks are described in Configure messaging bridges in *Administering Oracle WebLogic Server with Fusion Middleware Control*.

For information about configuring the messaging bridges that have been configured in this resource group, see [Configuring Messaging Bridges](#).

Configuring JMS Bridge Destinations

To configure the JMS bridge destination settings for a resource group:

1. Select the resource group that you want to configure.
2. Go to **Services > Messaging > Bridge Destinations**.

The following read-only information is available for the JMS bridge destinations configured in this resource group:

- Name
 - Adapter JNDI name
3. Create or delete JMS bridge destinations as needed.
 4. Save your changes.

These tasks are described in Configure JMS bridge destinations in *Administering Oracle WebLogic Server with Fusion Middleware Control*.

Configuring Path Services

To configure the path services settings for a resource group:

1. Select the resource group that you want to configure.
2. Go to **Services > Messaging > Path Services**.

The following read-only information is available for the path services configured in this resource group:

- Name
 - Persistent store
3. Create or delete path services as needed.
 4. Save your changes.

These tasks are described in Configure path services in *Administering Oracle WebLogic Server with Fusion Middleware Control*.

For information about configuring existing path services, see [Configuring Path Services to Support Using Unit-of-Order with Distributed Destinations](#).

Configuring Resource Group Mail Session Settings

To view configuration settings for the mail sessions that have been created in this resource group:

1. Select the resource group that you want to configure.

2. Go to **Services > Mail**.

The following read-only information is available for mail sessions configured in this resource group:

- Name
- JNDI name

3. Create or delete mail sessions as needed.

4. Save your changes.

These tasks are described in WebLogic Server Mail Sessions in *Administering Oracle WebLogic Server with Fusion Middleware Control*.

Configuring Resource Group Persistent Store Settings

To view configuration settings for the persistent stores that have been created in this resource group:

1. Select the resource group that you want to configure.

2. Go to **Services > Persistent Stores**.

The following read-only information is available for the persistent stores configured in this resource group:

- Name
- Type

3. Create or delete persistent stores as needed.

4. Save your changes.

These tasks are described in WebLogic Server Persistent Stores in *Administering Oracle WebLogic Server with Fusion Middleware Control*.

For information about configuring existing persistent stores, see [Configuring JDBC or File Persistent Stores](#).

Configuring Resource Group Foreign JNDI Provider Settings

To view configuration settings for the foreign JNDI providers that have been created in this resource group:

1. Select the resource group that you want to configure.

2. Go to **Services > Foreign JNDI Providers**.

The following read-only information is available for the foreign JNDI providers configured in this resource group:

- Name
- Initial context factory
- Provider URL
- User
- Targets

3. Create or delete foreign JNDI providers as needed.

4. Save your changes.

These tasks are described in WebLogic Server Foreign JNDI Providers in *Administering Oracle WebLogic Server with Fusion Middleware Control*.

For information about configuring existing foreign JNDI providers, see [Configuring and Programming JNDI](#).

Configuring Resource Group Diagnostic System Module Settings

To view configuration settings for the diagnostic system modules that have been created in this resource group:

1. Select the resource group that you want to configure.
2. Go to **Services > Diagnostics**.

The following read-only information is available for the diagnostic system modules configured in this resource group:

- Name
 - Description
 - Targets
3. Create or delete diagnostic system modules as needed.
 4. Save your changes.

These tasks are described in WebLogic Server Diagnostics in *Administering Oracle WebLogic Server with Fusion Middleware Control*.

For information about configuring existing diagnostic system modules, see [Monitoring and Debugging Partitions](#).

Configuring Resource Group Targets

To specify the targets for this resource group:

1. Select the resource group that you want to configure.
2. Go to **Targets**.
3. Specify one or more virtual targets to which this resource group is targeted.

For important considerations when targeting a resource group to more than one virtual target, see [Targeting a Resource Group to More Than One Target](#).

A virtual target can be used by many resource groups within a partition, or by many resource groups at the domain level.

4. Save your changes.

These tasks are described in Configure virtual targets in *Administering Oracle WebLogic Server with Fusion Middleware Control*.

For information about configuring the actual virtual target, see [Configuring Virtual Targets](#).

Configuring Resource Group Notes

To create notes for a resource group:

1. Select the resource group that you want to configure.

2. Go to **Notes**.
3. Enter your notes.
4. Save your changes.

Configuring Resource Groups: WLST Example

The example does the following:

1. Creates a domain partition.
2. Creates a virtual target.
3. Sets the host name and URI prefix for the virtual target.
4. Adds the virtual target as an available target in the partition.
5. Creates the resource group.
6. Adds the virtual target to the resource group.
7. Activates the changes.
8. Starts the partition.

Note:

If this is the first partition created in production mode, then you must restart the Administration Server before you can start the partition.

9. Deploys the application `MySimpleEjb` to the resource group.

```
# Create Pep partition and ResourceGroup
edit()
startEdit()
wls:/base_domain/edit/ !> domain=getMBean('/')
wls:/base_domain/edit/ !> peppart=domain.createPartition('Pep')
wls:/base_domain/edit/ !> vt=domain.createVirtualTarget('TestVT')
wls:/base_domain/edit/ !> vt.setHostNames(jarray.array([String('localhost')],String))
wls:/base_domain/edit/ !> vt.setUriPrefix('/foo')
wls:/base_domain/edit/ !> peppart.addAvailableTarget(vt)
wls:/base_domain/edit/ !> peprg=peppart.createResourceGroup('TestRG')
wls:/base_domain/edit/ !> peprg.addTarget(vt)
wls:/base_domain/edit/ !> activate()

wls:/base_domain/edit/ !> startPartitionWait(peppart)

wls:/base_domain/edit/ !> deploy(appName='MySimpleEjb', path='c:/webservices/
MySimpleEjb.jar', partition='Pep', resourceGroup='TestRG',
deploymentOrder=10,securityModel='DDOnly')
:
Completed the deployment of Application with status completed
Current Status of your Deployment:
Deployment command type: deploy
Deployment State : completed
Deployment Message : no message
```


Configuring Resource Groups: REST Example

For an example of configuring resource groups using REST, see *Creating Partitions in Administering Oracle WebLogic Server with RESTful Management Services*.

In particular, see the sections "View the new partition's resource groups" and "Demonstrate a partition deployer configuring system resources."

Deleting Resource Groups: Main Steps and WLST Example

You must first stop a resource group before you can delete it. Stopping the resource group causes applications and resources in the resource group to cease operating and to be removed from memory.

1. Select the resource group that you want to delete.
2. Go to **Control > Stop**. You should generally choose to stop the resource group when work completes.
3. Stop the resource group.
4. Delete the resource group.

Deleting Resource Groups: WLST Example

The example does the following:

1. Creates a domain partition.
2. Creates a virtual target.
3. Sets the host name and URI prefix for the virtual target.
4. Adds the virtual target as an available target in the partition.
5. Creates the resource group.
6. Adds the virtual target to the resource group.
7. Unsets the resource group.
8. Deletes the resource group.
9. Creates a different resource group.
10. Adds the virtual target to that resource group.
11. Activates the changes.
12. Starts the partition.



Note:

If this is the first partition created in production mode, then you must restart the Administration Server before you can start the partition.

```
# Create Pep partition and Resource Group. Remove Resource Group
edit()
startEdit()
```

```

wls:/base_domain/edit/ !> domain=getMBean('/')
wls:/base_domain/edit/ !> peppart=domain.createPartition('Pep')
wls:/base_domain/edit/ !> vt=domain.createVirtualTarget('TestVT')
wls:/base_domain/edit/ !> vt.setHostNames(jarray.array([String('localhost')],String))
wls:/base_domain/edit/ !> vt.setUriPrefix('/foo')
wls:/base_domain/edit/ !> peppart.addAvailableTarget(vt)
wls:/base_domain/edit/ !> peprg=peppart.createResourceGroup('TestRG')
wls:/base_domain/edit/ !> peprg.addTarget(vt)
wls:/base_domain/edit/ !> peppart.unSet('ResourceGroups')
wls:/base_domain/edit/ !> peprg=peppart.destroyResourceGroup(peprg)
wls:/base_domain/edit/ !> peprg=peppart.createResourceGroup('TestRG2')
wls:/base_domain/edit/ !> peprg.addTarget(vt)
wls:/base_domain/edit/ !> activate()
wls:/base_domain/edit/ !> startPartitionWait(peppart)

```

Controlling Resource Groups: Main Steps and WLST Example

Starting a resource group causes application deployments and resources that are not currently running to become active. Stopping a resource group causes applications and resources in the resource group to cease operating and to be removed from memory. It is the runtime equivalent to undeploying the application or resource, except that the configuration for the application or resource is not removed from the `config.xml` file as it would be in a true undeployment.

To control a resource group:

1. Select the resource group that you want to configure.
2. Go to **Control**.

You can perform the following actions:

- Start
- Stop

Controlling Resource Groups: WLST Example

Stopping a Resource Group

The following example builds on the WLST example shown in [Creating Resource Groups: WLST Example](#). It shows navigation to the `ResourceGroupLifeCycleRuntime` MBean and stops the resource group `TestRG`.

```

wls:/base_domain/serverConfig/> domainRuntime()
wls:/base_domain/domainRuntime/> domain=cmo
wls:/base_domain/domainRuntime/> partrun = cmo.lookupDomainPartitionRuntime('Pep')
wls:/base_domain/domainRuntime/> partliferun = partrun.getPartitionLifeCycleRuntime()
wls:/base_domain/domainRuntime/> rgliferun =
partliferun.lookupResourceGroupLifeCycleRuntime('TestRG')
wls:/base_domain/domainRuntime/> rgliferun.shutdown()
[MBeanServerInvocationHandler]com.bea:Name=_2_SHUTDOWN,Type=ResourceGroupLifeCyc
leTaskRuntime,DomainPartitionRuntime=Pep,ResourceGroupLifeCycleRuntime=TestRG,Pa
rtitionLifeCycleRuntime=Pep

```

Starting a Resource Group

The following example builds on the WLST example shown in [Creating Resource Groups: WLST Example](#). It shows navigation to the `ResourceGroupLifeCycleRuntime` MBean and starts the resource group `TestRG`.

```
wls:/base_domain/serverConfig/> domainRuntime()
wls:/base_domain/domainRuntime/> domain=cmo
wls:/base_domain/domainRuntime/> partrun = cmo.lookupDomainPartitionRuntime('Pep')
wls:/base_domain/domainRuntime/> partliferun = partrun.getPartitionLifeCycleRuntime()
wls:/base_domain/domainRuntime/> rgliferun =
partliferun.lookupResourceGroupLifeCycleRuntime('TestRG')
wls:/base_domain/domainRuntime/> rgliferun.start()
[MBeanServerInvocationHandler]com.bea:Name=_4_START,Type=ResourceGroupLifeCycleT
askRuntime,DomainPartitionRuntime=Pep,ResourceGroupLifeCycleRuntime=TestRG,Parti
tionLifeCycleRuntime=Pep
wls:/base_domain/domainRuntime/>
```

Migrating Resource Groups: Main Steps and WLST Example

When you migrate a resource group, you change the virtual target used by the resource group from one physical target (cluster or server) to another. After migration, the virtual target points to the new physical target (cluster or server). Note that this change affects any partition-level or domain-level resource group that uses this virtual target.

To use Fusion Middleware Control to migrate a resource group:

1. From the **WebLogic Domain** menu, select **Environment**, then select **Resource Groups**.
The Resource Groups table displays information about each resource group that has been configured in the current domain.
2. In the Resource Groups table, select the resource group that you want to configure.
3. Go to **Migrate**.
4. Choose a new target for the virtual target associated with the resource group. You can choose a single Managed Server or a single cluster.
5. Save your changes.

Migrating Resource Groups: WLST Example

The example does the following:

1. Creates a domain partition.
2. Creates a virtual target.
3. Sets the host name and URI prefix for the virtual target.
4. Targets the virtual target to the Administration Server. (You would typically not target the virtual target to the Administration Server.)
5. Adds the virtual target as an available target in the partition.

6. Creates the resource group.
7. Adds the virtual target to the resource group.
8. Activates the changes.
9. Starts the partition.

 **Note:**

If this is the first partition created in production mode, then you must restart the Administration Server before you can start the partition.

10. Removes the Administration Server as a target.
11. Migrates (targets) the virtual target to Cluster-0.

```
# Create Pep partition and ResourceGroup
edit()
startEdit()
wls:/base_domain/edit/ !> domain=getMBean('/')
wls:/base_domain/edit/ !> peppart=domain.createPartition('Pep')
wls:/base_domain/edit/ !> vt=domain.createVirtualTarget('TestVT')
wls:/base_domain/edit/ !> vt.setHostNames(jarray.array([String('localhost')],String))
wls:/base_domain/edit/ !> vt.setUriPrefix('/foo')
wls:/base_domain/edit/ !> tgt=getMBean('/Servers/AdminServer')
wls:/base_domain/edit/ !> vt.addTarget(tgt)
wls:/base_domain/edit/ !> peppart.addAvailableTarget(vt)
wls:/base_domain/edit/ !> peprg=peppart.createResourceGroup('TestRG')
wls:/base_domain/edit/ !> peprg.addTarget(vt)
wls:/base_domain/edit/ !> activate()
wls:/base_domain/edit/ !> startPartitionWait(peppart)
startEdit()
wls:/base_domain/edit/ !> vt.removeTarget(tgt)
wls:/base_domain/edit/ !> tgt=getMBean('/Clusters/Cluster-0')
wls:/base_domain/edit/ !> vt.addTarget(tgt)
wls:/base_domain/edit/ !> activate()
```

 **Note:**

During non-live migration, the data source used for the partition job scheduler service should be configured in an *administrative resource group*, meaning that it is not shut down when the partition is shutdown, before the migration. For more information about administrative resource groups, see [Managing Administrative Applications and Resources in Partition Resource Groups](#).

Managing Administrative Applications and Resources in Partition Resource Groups

In addition to application code that runs on Managed Servers and clusters, many applications and services include components that are considered *administrative*.

Administrative components primarily run on the Administration Server, and even when a partition is shutdown, it is required that these components continue to run.

An administrative application manages some supporting infrastructure used by one or more end-user applications. Typically, end users do not have access to administrative applications. An example might be a corporate product and part number catalog maintenance application. Internal users would update the catalog as products come and go, while end-user applications would refer to the catalog as they run end-user transactions. The JDBC system resource that the application uses to access the database where the catalog data is stored would be an example of an administrative resource.

If you designate the resource group containing administrative applications and resources an *administrative resource group*, then when you boot the partition (moving it from the `SHUTDOWN.HALTED` to the `SHUTDOWN.BOOTED` state) those applications will be available but none of the non-administrative, end-user-facing applications will be. For more information about the partition lifecycle states, see [About Partition Lifecycle States and Transitions](#).

Administrative resource groups:

- Contain administrative applications and resources.
- Are usually targeted to the Administration Server.
- Are handled in a unique manner by partition life cycle operations. See [About Partition Lifecycle States and Transitions](#).

In this release, you can:

- Designate applications and resources as administrative by placing them in an administrative resource group.
- Keep administrative applications and resources running when the partition is shutdown.
- Target administrative applications and resources to the Administration Server automatically. See [Targeting Administrative Resource Groups](#).

Creating Administrative Resource Groups

All applications and resources in a partition administrative resource group (whether defined directly in the resource group or indirectly through a resource group template) are considered administrative applications and resources.

To create an administrative resource group, set the resource group `Administrative` attribute to `true`.

Assuming that you have a WLST edit session in progress and `rg` is a variable containing the `ResourceGroupMBean` of interest:

```
rg.setAdministrative(true)
```

You may have zero or more administrative resource groups in a partition.

Targeting Administrative Resource Groups

You can target administrative resource groups like any other resource group, to a virtual target or the partition administrative virtual target, assuming it is available. See [About Partition Administrative Virtual Targets](#).

In this release, resource groups have a new Boolean attribute, `autoTargetAdminServer`, that by default, is false. When system administrators set this attribute to true, they target the resource group to the Administration Server, the same as explicitly adding the partition administrative virtual target as a resource group target.

Assuming that you have a WLST edit session in progress and `rg` is a variable containing the `ResourceGroupMBean` of interest:

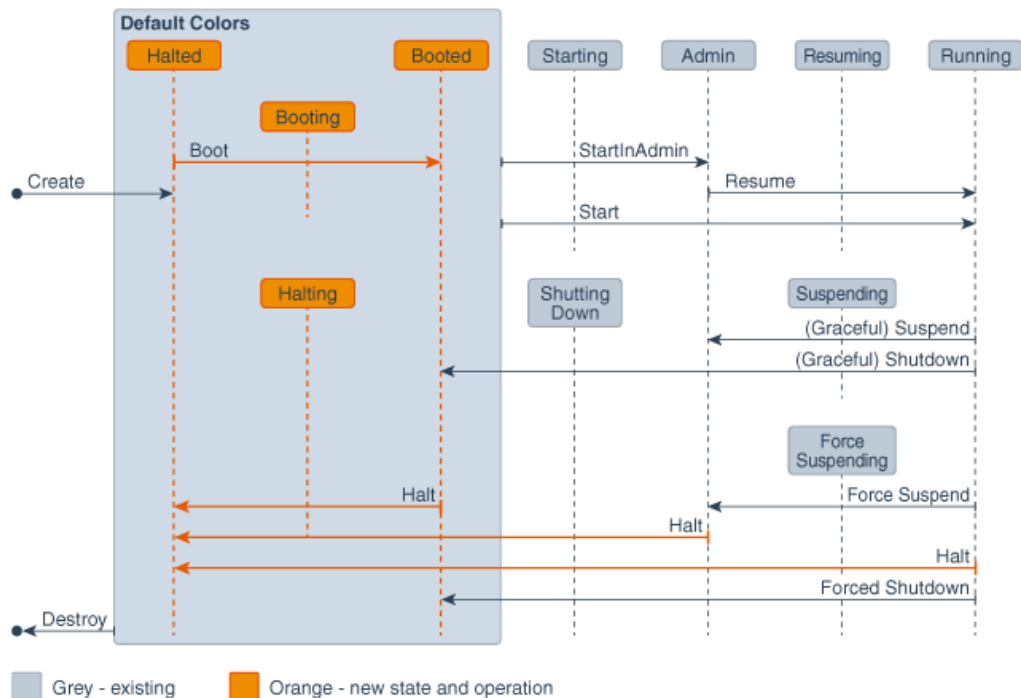
```
rg.setAutoTargetAdminServer(true)
```

For more information about partition administrative virtual targets, see [About Partition Administrative Virtual Targets](#).

About Partition Lifecycle States and Transitions

The partition lifecycle states are shown in [Figure 8-1](#). In this release, the partition lifecycle states are unchanged, but two substates have been added to the SHUTDOWN state: BOOTED and HALTED.

Figure 8-1 Partition Lifecycle States



Shutting down a partition takes it to the `SHUTDOWN.BOOTED` state. When a partition is in the `SHUTDOWN.BOOTED` state, administrative resource groups are running, but all other resource groups are shutdown.

When a partition is in the `SHUTDOWN.HALTED` state, all resource groups in the partition are shutdown and the partition is fully shutdown on all targets.

Newly created partitions are created in the `SHUTDOWN.HALTED` state. You can start a partition from either the `SHUTDOWN.BOOTED` or the `SHUTDOWN.HALTED` state.

The following table shows one possible workflow of partition lifecycle states going from partition creation to running.

Operation	Partition State	Partition Substate	Resource Group Activity
Create partition	SHUTDOWN	HALTED	NA
Create administrative resource groups and non-administrative resource groups	SHUTDOWN	HALTED	No resource groups are running
Target partition and resource groups	SHUTDOWN	HALTED	No resource groups are running
Start partition	RUNNING	NA	All resource groups are running
Shut down partition	SHUTDOWN	BOOTED	Only administrative resource groups are running
Halt partition	SHUTDOWN	HALTED	No resource groups are running

Some partition configuration changes require a partition restart to have them applied. That is typically done by shutting down the partition and then restarting it. For changes to administrative resource groups, the partition must be halted and then started or the administrative resource group must individually be restarted.

The following new lifecycle operations are supported on a partition:

- `halt()`: takes a partition from the `SHUTDOWN.BOOTED`, `ADMIN`, or `RUNNING` state to the `SHUTDOWN.HALTED` state.
- `boot()`: takes a partition from the `SHUTDOWN.HALTED` state to the `SHUTDOWN.BOOTED` state.

In summary:

- Booting a partition takes it from the `SHUTDOWN.HALTED` state to the `SHUTDOWN.BOOTED` state. Only the administrative applications are running. Partition administrators can manage the partition but end users cannot access the end-user-facing applications (because they are not running).
- Starting a partition takes it from the `SHUTDOWN` state (either the `SHUTDOWN.HALTED` or the `SHUTDOWN.BOOTED` state) to running. All applications are running.
- Shutting down a partition takes it from the `RUNNING` or `ADMIN` state to the `SHUTDOWN.BOOTED` state. Administrative applications are running. Partition administrators can manage the partition; end users cannot access applications.

- Halting a partition takes it from the `RUNNING` or `ADMIN` state or the `SHUTDOWN,BOOTED` state to the `SHUTDOWN,HALTED` state. No applications are running in the partition. Partition administrators cannot manage the partition.

Configuring Resource Groups: Related Tasks and Links

- [Oracle WebLogic Server Multitenant](#)
- [Configuring Virtual Targets](#)
- [Configuring Resource Group Templates](#)
- [Configuring Resource Overrides](#)
- [Configuring Messaging](#)

9

Configuring Resource Overrides

Resource overriding allows administrators to customize resources at the partition level. You can use Oracle Enterprise Manager Fusion Middleware Control (FMWC), the Oracle WebLogic Server Administration Console, or WebLogic Scripting Tool (WLST) to configure resource overrides.

 **Note:**

WebLogic Server Multitenant domain partitions, resource groups, resource group templates, virtual targets, and Resource Consumption Management are deprecated in WebLogic Server 12.2.1.4.0 and will be removed in the next release.

The following information describes resource overrides and how to configure them.

- [Resource Overrides: Overview](#)
- [Configuring Resource MBean Overrides: Main Steps and WLST Examples](#)
- [Configuring Resource Deployment Plans: Main Steps and WLST Example](#)
- [Configuring Resource Overrides: Related Tasks and Links](#)

Resource Overrides: Overview

If you create a partition with a resource group that extends a resource group template, then you can override settings for certain resources defined in that resource group template.

 **Note:**

Resource override configuration MBeans are deprecated in WebLogic Server 12.2.1.4.0 and will be removed in the next release.

If you create a resource group within the partition that does *not* extend a resource group template and create resources within this resource group, then you do not need overrides; you can just set partition-specific values for these resources. Overrides are used mainly when there is a common definition for the resource, such as in a resource group template.

Resource group templates are particularly useful in Software-as-a-Service (SaaS) environments where WebLogic Server Multitenant (MT) activates the same applications and resources multiple times, once per domain partition. Some of the information about such resources is the same across all domain partitions, while some of it, such as JMS queues and database connections, varies from one partition to the

next. For example, you would need to customize the URL, user name, and password used to connect to a data source among different partitions.

Administrators can override resource definitions in partitions using the following techniques:

- Resource override configuration MBeans: A configuration MBean that exposes a subset of attributes of an existing resource configuration MBean. Any attribute set on an instance of an overriding configuration MBean replaces the value of that attribute in the corresponding resource configuration MBean instance.
- Resource deployment plans: An XML file that identifies resources within a partition and overrides attribute settings on those resources.
- Partition-specific application deployment plans: Similar to application deployment plans, it allows administrators to specify a partition-specific application deployment plan for each application deployment in a partition. For information about partition-specific application deployment plans, see [Using Partition-Specific Deployment Plans](#).

Administrators can combine any of these resource overriding techniques. The system applies them in the following, ascending order of priority:

- The `config.xml` file and external descriptors, including partition-specific application deployment plans
- Resource deployment plans
- Overriding configuration MBeans

If an attribute is referenced by both a resource deployment plan and an overriding configuration MBean, then the overriding configuration MBean takes precedence.

Using Configuration MBean Overrides

Overriding configuration MBeans lets you customize *frequently tailored* attributes, such as the attributes of a JMS queue or of a database connection. An attribute in an overriding configuration MBean might override the `config.xml` settings for the resource or it might override a setting from the resource's external descriptor, depending on where the attribute resides.

In this release, WebLogic Server MT provides the following resource override configuration MBeans:

- `JDBCSystemResourceOverrideMBean`
- `JMSSystemResourceOverrideMBean`
- `ForeignServerOverrideMBean`
- `ForeignConnectionFactoryOverrideMBean`
- `ForeignDestinationOverrideMBean`
- `MailSessionOverrideMBean`

Resource configuration override MBeans typically work by matching the name of the overriding configuration MBean with the name of the resource MBean. See [Configuring Resource MBean Overrides: Main Steps and WLST Examples](#).

Using Resource Deployment Plans

A resource deployment plan is an XML file that overrides attributes for one or more resources within a single partition. Each partition can have at most one resource deployment plan that can override any resources in that partition. This includes resources defined in resource group templates to which the partition's resource groups refer as well as resources declared directly in the resource groups.

When a server restarts, changes to nondynamic attributes from the resource deployment plan (and from overriding configuration MBeans), are applied before the resource is active. After a given resource is running, changes to the resource deployment plan that would affect nondynamic attributes of that resource do not take effect until the partition is restarted.

Resource deployment plans can be used with the following resources to override options that are configured in the `config.xml` file or external descriptor files:

- `CoherenceClusterSystemResource`
- `FileStore`
- `ForeignJNDIProvider`
- `JDBCStore`
- `JDBCSystemResource`
- `JMSBridgeDestination`
- `JMSServer`
- `JMSSystemResource`
- `MailSession`
- `ManagedExecutorService`
- `ManagedScheduledExecutorService`
- `ManagedThreadFactory`
- `MessagingBridge`
- `PathService`
- `SAFAgent`
- `WLDFSystemResource`

resource-deployment-plan Syntax

Resource deployment plans are based on the [weblogic-resource-deployment-plan.xsd](#) file and possess a similar syntax to WebLogic Server application deployment plans. Resource deployment plans identify the resource and attribute values to be changed; they support replacing values, adding new ones, and removing existing ones.

The following is a summary of the `resource-deployment-plan` syntax:

```
resource-deployment-plan (@global-variables)
  description
  version
```

```

variable-definition
  variable*
    name
    value
external-resource-override*
  resource-name
  resource-type
  root-element
  variable-assignment*
    name
    xpath
config-resource-override*
  resource-name
  resource-type
  variable-assignment*
    name
    xpath
    operation

```

The @ symbol indicates an XML attribute, and an asterisk (*) means the element can be repeated. The `operation` element is optional but allowed in any `variable-assignment` element, whether in an `external-resource-override` element or a `config-resource-override` element.

The basic elements in the resource deployment plan serve the following functions:

- `resource-deployment-plan` is the root element; it encapsulates all of the resource deployment plan's contents.
- `variable-definition` defines one or more variable elements. Each variable defines the name of a variable used in a plan and a value to assign (which can be null). The sample plan shown in [Sample Resource Deployment Plan](#) contains variable definitions for changes to the mail session user name, mail session JavaMail properties, and JDBC connection pool parameters, and test table name properties.
- Each `external-resource-override` and `config-resource-override` element (and their child elements) does these three things:

1. Identifies the resource to affect (`resource-name` and `resource-type`).

The resources in a partition must have unique names within the partition.

2. Identifies where the attributes are defined.

All the attributes in the `config-resource-override` element reside in the `config.xml` file. Attributes in the `external-resource-override` element are stored in external descriptor files. For resource deployment plans, the descriptor path is relative to the partition's `config/` directory.

3. Specifies some number of attributes of that resource to override (`variable-assignment` elements).

After the attributes for the resource are located, the `variable-assignment` elements are applied. An XPath expression tells where, relative to the identified resource's bean, the attribute to be affected appears in the bean tree. The name refers to a previously defined `variable` definition. That `variable` also sets the value that should replace whatever is in the original attribute setting.

By default, the values in `variable-assignment` elements are added to the values that are already defined in the descriptor. You can change this behavior

and cause the `variable-assignment` element to replace or remove the values that are defined in the descriptor by setting the `operation` subelement in the `variable-assignment` element to the value `replace` or `remove`, respectively.

For more information about the contents of a WebLogic Server resource deployment plan, see <http://xmlns.oracle.com/weblogic/resource-deployment-plan/1.0/resource-deployment-plan.xsd>. For more information about WebLogic Server application deployment plans, see *Understanding Deployment Plan Contents in Deploying Applications to Oracle WebLogic Server*.

Sample Resource Deployment Plan

The following shows a sample resource deployment plan:

```
<resource-deployment-plan
  xmlns="http://xmlns.oracle.com/weblogic/resource-deployment-plan"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema"
  xsi:schemaLocation="http://xmlns.oracle.com/weblogic/resource-deployment-plan
http://xmlns.oracle.com/weblogic/resource-deployment-plan/1.0/resource-deployment-
plan.xsd"
>
  <description>Example RDP</description>
  <variable-definition>
    <variable>
      <name>mailNewSessionUsername</name>
      <value>aaron</value>
    </variable>
    <variable>
      <name>mailNewProps</name>
      <value>mail.user=someoneElse;mail.host=myMailServer.oracle.com</value>
    </variable>
    <variable>
      <name>jdbcTableToTest</name>
      <value>JUNK_TABLE</value>
    </variable>
  </variable-definition>
  <external-resource-override>
    <resource-name>myjdbc</resource-name>
    <resource-type>jdbc-system-resource</resource-type>
    <root-element>jdbc-data-source</root-element>
    <variable-assignment>
      <name>jdbcTableToTest</name>
      <xpath>/jdbc-data-source/jdbc-connection-pool-params/test-table-name</xpath>
    </variable-assignment>
  </external-resource-override>
  <config-resource-override>
    <resource-name>myMail</resource-name>
    <resource-type>mail-session</resource-type>
    <variable-assignment>
      <name>mailNewSessionUsername</name>
      <xpath>/mail-session/session-username</xpath>
    </variable-assignment>
    <variable-assignment>
      <name>mailNewProps</name>
      <xpath>/mail-session/properties</xpath>
    </variable-assignment>
  </config-resource-override>
</resource-deployment-plan>
```

Configuring Resource MBean Overrides: Main Steps and WLST Examples

When configuring a resource MBean override, you define a system resource override with a name that matches the resource you want to override, then configure partition-specific attributes for it.

Prior to creating resource overrides in a domain partition, you must have first defined the resource that you want to override in a resource group template.

Configuring a JDBC System Resource Override: Main Steps

The main steps for configuring a JDBC system resource override are as follows:

1. Create the JDBC system resource override with the name of the data source that you want to override.
2. Optionally, provide partition-specific URL, user name, and password values.
3. If the resource is running, then restart the partition for the overrides to take effect.

Configuring a JDBC System Resource Override: WLST Example

The following example shows how to configure a JDBC system resource override using WLST:

```
edit()
startEdit()
cd('/Partitions/myPartition')
cmo.createJDBCSystemResourceOverride('myGDS')

cd('/Partitions/myPartition/JDBCSystemResourceOverrides/
myJDBCSystemResourceOverride')
cmo.setURL('jdbc:oracle:thin:@lcr01156:1521:wldevdb2')
cmo.setUser('admin')
setEncrypted('Password', 'Password_1440013198602', 'C:/Oracle/Middleware/Oracle_Home/
user_projects/domains/base_domain/Script1440013057535Config', 'C:/Oracle/Middleware/
Oracle_Home/user_projects/domains/base_domain/Script1440013057535Secret')
activate()
```

Configuring a JMS System Resource Override: Main Steps

The main steps for configuring a JMS system module or foreign server override are as follows:

1. Provide a name for the foreign server override.
The system matches overrides using the name of the overriding configuration MBean with the name of the actual resource to be overridden.
2. Optionally, provide partition-specific values for initial context factory, connection URL, JNDI properties credential, and JNDI properties.
3. If the resource is running, then restart the partition for the overrides to take effect.

The main steps for configuring a JMS foreign server or foreign destination override are as follows:

1. Provide a foreign JMS destination name for the foreign destination override.
The name of this foreign destination override must match the name of the actual resource to be overridden.
2. Optionally, provide a partition-specific remote JNDI name value.
3. If the resource is running, then restart the partition for the overrides to take effect.

The main steps for configuring a JMS foreign server or foreign connection factory override are as follows:

1. Provide a foreign JMS connection factory name for the foreign connection factory override.
The name of this foreign connection factory override must match the name of the actual resource to be overridden.
2. Optionally, provide partition-specific values for remote JNDI name, user name, and password.
3. If the resource is running, then restart the partition for the overrides to take effect.

Configuring a JMS System Resource Override: WLST Example

The following example shows how to configure a JMS system resource override using WLST:

```
edit()
startEdit()
cd('/Partitions/myPartition')
cmo.createJMSSystemResourceOverride('myJMSmoduleOverride')
activate()

startEdit()
cd('/Partitions/myPartition/JMSSystemResourceOverrides/myJMSmoduleOverride')
cmo.createForeignServer('newJMSforeignServer')

cd('/Partitions/myPartition/JMSSystemResourceOverrides/myJMSmoduleOverride/
ForeignServers/newJMSforeignServer')
setEncrypted('JNDIPropertiesCredential', 'JNDIPropertiesCredential_1440013308523',
'C:/Oracle/Middleware/Oracle_Home/user_projects/domains/base_domain/
Script1440013057535Config', 'C:/Oracle/Middleware/Oracle_Home/user_projects/domains/
base_domain/Script1440013057535Secret')
cmo.setConnectionURL('java.naming.provider.url')
cmo.setInitialContextFactory('weblogic.jndi.WLInitialContextFactory')
activate()

startEdit()
cmo.createForeignDestination('myForeignDestinationOverride')
cd('/Partitions/myPartition/JMSSystemResourceOverrides/myJMSmoduleOverride/
ForeignServers/newJMSforeignServer/ForeignDestinations/myForeignDestinationOverride')
cmo.setRemoteJNDIName('/remote')
activate()

startEdit()
cd('/Partitions/myPartition/JMSSystemResourceOverrides/myJMSmoduleOverride/
ForeignServers/newJMSforeignServer')
cmo.createForeignConnectionFactory('myJMSconnectionFactoryOverride')
```

```

cd('/Partitions/myPartition/JMSSystemResourceOverrides/myJMSmoduleOverride/
ForeignServers/newJMSForeignServer/ForeignConnectionFactory/
myJMSConnectionFactoryOverride')
setEncrypted('Password', 'Password_1440013370283', 'C:/Oracle/Middleware/Oracle_Home/
user_projects/domains/base_domain/Script1440013057535Config', 'C:/Oracle/Middleware/
Oracle_Home/user_projects/domains/base_domain/Script1440013057535Secret')
cmo.setUsername('wlsqa')
cmo.setRemoteJNDIName('/remote')
activate()

```

Configuring a Mail Session Override: Main Steps

The main steps for configuring a mail session override are as follows:

- Provide a name for the mail session override.
The system matches overrides using the name of the overriding configuration MBean with the name of the resource MBean.
- Optionally, provide partition-specific values for session user name, session password, and JavaMail properties.
- If the resource is running, then restart the partition for the overrides to take effect.

Configuring a Mail Session Override: WLST Example

The following example shows how to configure a mail session override using WLST:

```

edit()
startEdit()
cd('/Partitions/myPartition')
cmo.createMailSessionOverride('myMailSessionOverride')
cd('/Partitions/myPartition/MailSessionOverrides/myMailSessionOverride')
cmo.setSessionUsername('wlsqa')
setEncrypted('SessionPassword', 'SessionPassword_1440013452297', 'C:/Oracle/
Middleware/Oracle_Home/user_projects/domains/base_domain/Script1440013057535Config',
'C:/Oracle/Middleware/Oracle_Home/user_projects/domains/base_domain/
Script1440013057535Secret')
cmo.setProperties({})
activate()

```

Configuring Resource Deployment Plans: Main Steps and WLST Example

You configure a resource deployment plan by creating an XML document that identifies resources within a partition and overrides attribute settings on those resources.

1. Using an XML editor, create a resource deployment plan. See [resource-deployment-plan Syntax](#) and [Sample Resource Deployment Plan](#).
2. Associate the resource deployment plan with a partition by providing the path to the resource deployment plan file.
3. If the resource is running, then restart the partition.

The following shows the WLST commands for associating a resource deployment plan with an existing partition:

```
edit()
startEdit()
partition=cmo.lookupPartition(partitionName)
partition.setResourceDeploymentPlanPath('/path/to/the/file.xml')
save()
activate()
```

Configuring Resource Overrides: Related Tasks and Links

The referenced information provides steps for configuring resource overrides using Oracle Enterprise Manager Fusion Middleware Control (FMWC) and the Oracle WebLogic Server Administration Console.

- WebLogic Server Resource Overrides in *Administering Oracle WebLogic Server with Fusion Middleware Control*.
- JDBC Overrides, Mail Session Overrides, and JMS Overrides in *Oracle WebLogic Server Administration Console Online Help*.

10

Configuring Resource Consumption Management

You use resource consumption management (RCM) to ensure the fairness of resource allocation and reduce the contention of shared resources by collocated domain partitions in the server instance. Using Oracle Enterprise Manager Fusion Middleware Control (FMWC) or WebLogic Scripting Tool (WLST), you can create RCM policies that provide consistent performance of domain partitions in multitenant environments.

Note:

WebLogic Server Multitenant domain partitions, resource groups, resource group templates, virtual targets, and Resource Consumption Management are deprecated in WebLogic Server 12.2.1.4.0 and will be removed in the next release.

This chapter includes the following sections:

- [Configuring Resource Consumption Management: Overview](#)
- [Configuring Resource Consumption Management: Main Steps](#)
- [Dynamic Reconfiguration of Resource Managers](#)
- [Configuring Resource Consumption Management: Monitoring Resource Use](#)
- [Best Practices and Considerations When Using Resource Consumption Management](#)
- [RCM Limitations](#)
- [Configuring Resource Consumption Management: WLST Example](#)
- [Configuring Resource Sharing: Related Tasks and Links](#)

Configuring Resource Consumption Management: Overview

Resource consumption management (RCM) provides a flexible, dynamic mechanism for WebLogic Server system administrators to manage shared resources and provide consistent performance of domain partitions in MT environments.

RCM policies are configured as resource managers. A resource manager can be created with a global scope at the domain level and then used as the RCM policy for any partition within the domain. You can also create a partition-scoped resource manager if the partition has RCM characteristics specific to that partition. See [Configuring Resource Consumption Management: Main Steps](#).

Software Requirements for Using RCM

RCM requires Oracle Java Development Kit (JDK) 8u60 build 32 or later.

Why Do You Need Resource Consumption Management?

When applications are deployed to multiple domain partitions, sharing low-level resources such as CPU, heap, network, and file descriptors can result in unfair resource allocation. Unusual resource consumption requests may happen for various reasons such as high traffic, application design, or malicious code. These request types can overload the capacity of a shared resource, preventing another collocated domain partition's access to the resource. By employing appropriate RCM policies at the domain partition level, resource consumption management prevents applications in one partition from negatively affecting applications in other partitions.

How to Enable RCM

Set the following JVM arguments to enable RCM in your environment:

```
-XX:+UnlockCommercialFeatures -XX:+ResourceManagement -XX:+UseG1GC
```

This flag must be applied on every server instance (JVM) where RCM is enabled.

An alternative method is to uncomment these `JAVA_OPTIONS` in the `startWeblogic.sh` file which gets created when a domain is created:

```
#JAVA_OPTIONS="-XX:+UnlockCommercialFeatures -XX:+ResourceManagement -XX:+UseG1GC $  
{SAVE_JAVA_OPTIONS}
```

You must do this on every server instance, and it must be done prior to starting the WebLogic Server instance.

If a Java Security Manager is used with WebLogic Server, then RCM requires the granting of the following permission in `weblogic.policy`:

```
permission RuntimePermission("jdk.management.resource.getResourceContextFactory")
```

For more information about using the Java Security Manager to protect resources in Weblogic Server, see [Using the Java Security Manager to Protect WebLogic Resources in *Developing Applications with the WebLogic Security Service*](#).

Supported Resources for RCM

The following shared resources can be managed through RCM policies:

- `FileOpen`: The number of open file descriptors in use by a partition. This includes files opened through `FileInputStream`, `FileOutputStream`, `RandomAccessFile`, and NIO File channels.
- `HeapRetained`: The amount of heap (in MB) retained or in use by a partition.
- `CpuUtilization`: The percentage of CPU time used by a partition with respect to the available CPU time of the WebLogic Server process.

Configuring Resource Consumption Management: Main Steps

A system administrator specifies resource consumption management policies on shared resources for each partition in the domain using a resource manager. A resource manager consists of one or more policies for one or more shared resources. Each policy consists of a constraint value for a resource and a specified action a WebLogic Server instance takes when the constraint value is met.

The following RCM policy types are supported:

- [Triggers](#)
- [Fair Share](#)

Triggers

A trigger defines the static constraint value for the allowed resource use. When the consumption of that resource exceeds the constraint value, a specified action is performed. This policy type is best suited for environments where the resource use by a partition in the domain is predictable.

A system administrator can select the following actions when creating a trigger policy:

- **Notify:** Provides notification to the system administrator that the constraint value has been reached. You can add more than one `Notify` trigger for a resource. For example, `Notify when the Open Files go beyond 20`; `Notify when the Open Files go beyond 50`. Also, you can use the WebLogic Diagnostics Framework (WLDF) to create watch rules to listen to log messages and provide advanced notifications.
- **Slow:** Slows the rate at which the resource is consumed. When a `Slow` action is triggered, the Partition Work Manager's fair share value is reduced, which results in reducing the `thread-usage` time made available to the partition. For more information about Partition Work Managers, see [Configuring Partition Work Managers](#).
- **Fail:** Fails resource consumption requests for a resource until use is below the constraint value.

 **Note:**

Fail is applicable only for `Open Files` and not for other resources.

For example, to limit the number of open files in partition P1 to fewer than 100 files, create a resource manager with a trigger policy that has a constraint value of 100 units for the `Open Files` resource and a `Fail` action for the P1 partition.

- **Shutdown:** Initiates the shutdown of a partition while allowing cleanup. This action is useful when a partition exceeds a known constraint value and an adverse effect of shared resources used by other partitions in the domain is expected. A partition is only shut down in the Managed Server when the constraint value has been met,

allowing continuous availability in clustered environments. Fail and Shutdown triggers should not be used together.

- **Restart:** Restarts the partition on the server instance on which the partition's resource consumption quotas have been breached. Many times an application misbehaves temporarily and a simple restart will suffice to correct the situation. Intervention by a system administrator would be required, but less desirable than enabling this partition auto-restart action. Like other RCM actions, the partition restart action is scoped to a given Managed Server.

Additionally, you may specify the following optional configuration properties to prevent a partition from restarting in a loop.

- `max-restart-allowed`

The maximum number of RCM initiated partition restarts allowed in the specified time interval, after which the partition will be halted upon an RCM initiated request to restart the partition.

- `max-restart-allowed-interval`

The `max-restart-allowed-interval` is a fixed-width, sliding-window time interval (in seconds) within which the specified number (in `max-restart-allowed`) of RCM initiated partition restarts are permitted. A request to restart the partition that exceeds the `max-restart-allowed` number in the `max-restart-allowed-interval` leads to the partition being halted rather than being restarted.

- `restart-delay`

A delay (in seconds) introduced before starting the partition.

You can define these properties in the RCM policy, under `<resource-manager>` in the `config.xml` file. All the restart actions within a given RCM policy share the values for these configurable properties. Administrators can also provide a restart delay, when the partition is being restarted. The configured delay would be introduced before starting the partition, and would avoid the partition being restarted in a tight loop or restarting too soon for some temporary external error condition to clear.

Below is a sample configuration for the RCM restart action.

```
<resource-manager>
  <name>ResourceManager-1</name>

  <restart-loop-protection>

    <!--This indicates whether restart loop protection is enabled or
      not. If you want to disable the restart loop protection,
      set this flag to false. -->
    <restart-loop-protection-enabled>true</restart-loop-protection-enabled>

    <!-- The partition can be restarted a maximum of 3 times in 60 minutes
      by the RCM restart action. Subsequently, the partition
      would be halted. -->
    <max-restart-allowed>3</max-restart-allowed>

    <!-- Within any contiguous interval of 60 minutes,
      a maximum number of 3 restarts are allowed,
      as specified by the max-restart-allowed property. -->
    <max-restart-allowed-interval>3600</max-restart-allowed-interval>
```

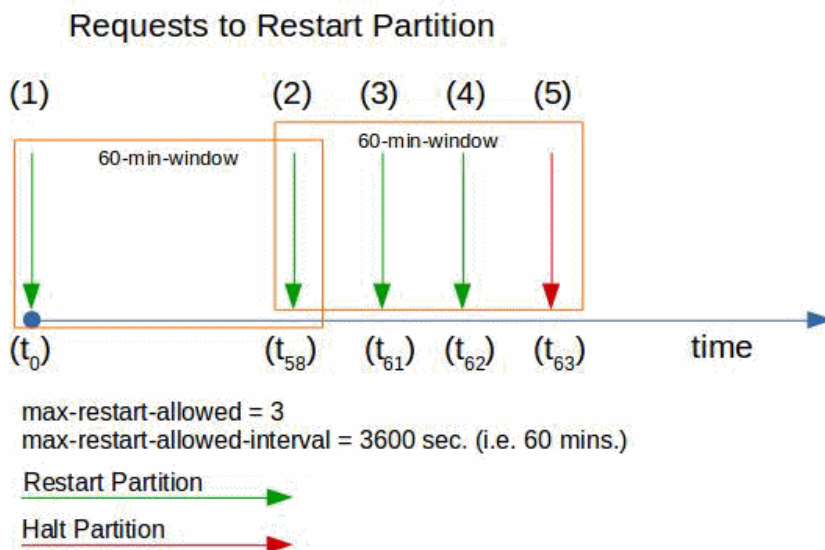
```

    <!-- Introduce a delay of 10 seconds before starting the
         partition. -->
    <restart-delay>10</restart-delay>
</restart-loop-protection>

<file-open>
  <name>RM1-FileOpenResource</name>
  <trigger>
    <name>RM1-Trigger-1</name>
    <value>20</value>
    <action>restart</action>
  </trigger>
</file-open>
</resource-manager>

```

When the configured triggers for a restart action are breached, a request to restart the partition is generated, as shown in the following illustration.



Note:

Policy actions may be implemented synchronously or asynchronously depending on the action type. For instance, the `Fail` action synchronously uses the thread that requested the file open. Other actions, such as the `Slow` action configured for a `Heap Retained` resource proceed asynchronously.

Fair Share

The fair share policy allows a system administrator to allocate a share (a percentage of the available resource) based on a representative load of each partition in the domain. Fair share policies are used when the exact use requirements of a resource

cannot be determined or are not practical to implement when using resource managers to provide the efficient and fair use of resources. When there is no contention between partitions in a domain for a given resource, each partition is able to use the amount of resources required for its immediate workload. If there is contention between partitions for a given resource, then each partition is constrained to use only its fair share of the available resource. Ensure that limits are set such that overall memory consumption does not cross the maximum available memory resulting in an out-of-memory exception.

Determining Fair Share Allocations for a Resource

A share is an allocation to use a specified amount of a resource. A system administrator allocates a share to a partition by specifying an integer value between 1 and 1000 in the associated resource manager fair share policy. For a given partition, the ratio of its configured fair share value to the sum total of all fair share values for the same resource in the domain determines the amount of resource allocated.

For example, a system administrator specifies a fair share value of 150 for a resource in partition P1 and a value of 100 for the same resource partition P2. If the workload is heavy enough in both partitions to create contention for that resource, then the resource allocation for partitions P1 is $150/(150+100)$ or 60 percent of the available resource.

Creating a Resource Manager

A resource manager consists of one or more policies for one or more shared resources. Each policy consists of a constraint value for a resource and a specified action that a WebLogic Server instance takes when the constraint value is met.

To configure resource managers, see:

- Create resource managers in *Administering Oracle WebLogic Server with Fusion Middleware Control*.
- [Configuring Resource Consumption Management: WLST Example](#)

Example RCM Configuration in config.xml

The following example is an annotated RCM configuration similar to the WLST example displayed in [Configuring Resource Consumption Management: WLST Example](#).

```
<domain>
...
  <!--Define RCM Configuration -->
  <resource-management>
    <resource-manager>
      <name>Approved</name>
      <file-open>
        <trigger>
          <name>Approved2000</name>
          <value>2000</value><!-- in units-->
          <action>shutdown</action>
        </trigger>
        <trigger>
          <name>Approved1700</name>
          <value>1700</value>
```

```

        <action>slow</action>
    </trigger>
</trigger>
    <name>Approved1500</name>
    <value>1500</value>
    <action>notify</action>
</trigger>
</file-open>
<heap-retained>
    <trigger>
        <name>Approved2GB</name>
        <value>2097152</value>
        <action>shutdown</action>
    </trigger>
    <fair-share-constraint>
        <name>FS-ApprovedShare</name>
        <value>60</value>
    </fair-share-constraint>
</heap-retained>
</resource-manager>

<resource-manager>
    <name>Trial</name>
    <file-open>
        <trigger>
            <name>Trial1000</name>
            <value>1000</value><!-- in units-->
            <action>shutdown</action>
        </trigger>
        <trigger>
            <name>Trial700</name>
            <value>700</value>
            <action>slow</action>
        </trigger>
        <trigger>
            <name>Trial500</name>
            <value>500</value>
            <action>notify</action>
        </trigger>
    </file-open>
    ...
</resource-manager>

<resource-manager>
    <name>RestartPartition</name>
    <file-open>
        <name>FileOpenQuota</name>
        <trigger>
            <name>MaxFileOpenAllowed</name>
            <value>20</value>
            <action>restart</action>
        </trigger>
    </file-open>
    <restart-loop-protection>
        <restart-loop-protection-enabled>true</restart-loop-protection-
enabled>
        <max-restart-allowed>3</max-restart-allowed>
        <max-restart-allowed-interval>3600
        </max-restart-allowed-interval>
        <restart-delay>10</restart-delay>
    </restart-loop-protection>

```



```

        </resource-manager>
    </resource-management>

    <partition>
        <name>Partition-0</name>
        <resource-group>
            <name>ResourceTemplate-0_group</name>
            <resource-group-template>ResourceTemplate-0</resource-group-template>
        </resource-group>
        ...
        <partition-id>1741ad19-8ca7-4339-b6d3-78e56d8a5858</partition-id>

        <!-- RCM Managers are targeted to partitions during partition
             creation or later by system administrators. -->
        <resource-manager-ref>Approved</resource-manager-ref>
        ...
    </partition>
    ..
</domain>

```

Dynamic Reconfiguration of Resource Managers

You can dynamically apply or remove a resource management policy from a domain partition. Changes to a resource management policy will be applied to all domain partitions that use that policy.

If a policy update for an active domain partition sets trigger values for a resource that is lower than the current use of that resource, then subsequent use of that resource would have the policy's recourse action applied. If a change to a policy would result in an immediate shutdown of an active domain partition based on the current use value, then the change would not be accepted as a dynamic reconfiguration change.

Configuring Resource Consumption Management: Monitoring Resource Use

You use resource consumption management metrics to monitor the current resource use in a partition, and to profile and analyze the resource consumption of a partition to generate data such as representative loads, peak loads, and peak load times needed to create effective resource managers and WLDF watches and notifications. RCM metrics for shared resources in a partition are available through a [PartitionResourceMetricsRuntimeMBean](#).

To monitor resource managers in Fusion Middleware Control, see Monitor resource managers in *Administering Oracle WebLogic Server with Fusion Middleware Control*.

By default, eager registration of resource meters is turned off. As a result, they get created lazily the first time the resource consumption metrics are queried for a particular resource. In that case, where the resource accounting is started lazily, the values returned from the resource consumption metrics might be different from the actual values of the resource consumed by the partition.

To get a true reflection of the amount of a resource consumed by a partition, the meters should be registered eagerly on partition startup. To enable eager registration of the resource meters, when starting the WebLogic Server instance, set the property, `weblogic.rcm.enable-eager-resource-meter-registration`, to `true`, as a JVM argument.

Best Practices and Considerations When Using Resource Consumption Management

Best practices for system administrators developing resource management policies include monitoring average and peak resource use data and carefully selecting recourse actions, such as deciding when to use triggers, fair share policies, complementary workloads, and more.

- [General Considerations](#)
- [Monitor Average and Peak Resource Use](#)
- [When to Use a Trigger](#)
- [When to Use Fair Share](#)
- [Use Complementary Workloads](#)
- [When to Use Partition-Scoped RCM Policies](#)
- [Managing CPU Use](#)
- [Managing Heap](#)

General Considerations

Recourse actions must be selected carefully by a system administrator. A lot of resources have complex interactions between them. For instance, slowing down CPU use (resulting in fewer threads allocated to the domain partition) may result in increased heap residency, thereby affecting retained heap use.

For a slow recourse action to be effective, applications must not create or manage threads. Oracle recommends that applications use any of the WebLogic Server capabilities like EJB Timers, Common J Work Manager and Timers, Managed Executor Service, Batch API and such, to manage the tasks, so that the slow recourse action will be effective.

Monitor Average and Peak Resource Use

Before specifying RCM policies, Oracle recommends that system administrators monitor average and peak resource use data and configure policies with sufficient headroom to balance efficient use of resources and meeting their Service Level Agreements (SLAs). See [Configuring Resource Consumption Management: Monitoring Resource Use](#).

When to Use a Trigger

Use triggers when an administrator is aware of precise limits at which the corresponding trigger needs to be executed. The trigger will be executed after the configured threshold is exceeded for some resources like file, and may be delayed for some of the resources like heap and CPU.

When to Use Fair Share

A fair share policy is typically used by a system administrator to ensure that a bounded-size shared resource is shared effectively (and fairly) by competing consumers. A fair share policy may also be employed by a system administrator when a clear understanding of the exact use of a resource by a partition cannot be accurately determined in advance, and the system administrator would like efficient use of resources while ensuring fair allocation of shared resources to co-resident partitions. Use fair share policies in your environment when you have complementary workloads for a resource between partitions. See [Use Complementary Workloads](#).

Use Complementary Workloads

When possible, maximize resource density by balancing the peak use times between partitions so that there is no overlap in peak use times and the sum of their averages is not above their maximum peak value. Antagonistic workloads on the other hand have overlapping peak use times and their sum of averages is greater than their maximum peak values.

Also consider collocating partition workloads that exercise resources differently. For instance, hosting a partition that has a predominantly CPU-bound workload with another partition that has a memory-bound workload could help in achieving better density and improving overall resource use.

When to Use Partition-Scoped RCM Policies

Use partition-scoped dynamic RCM policies if a partition has unique resource requirements. These policies facilitate easy import and export of partition RCM policies to and from existing domains.

If no resource management policies are explicitly set on a partition, then that partition has unconstrained access to available shared resources.

Managing CPU Use

CPU use is an excellent metric to track contention of CPU by collocated domain partitions, and is especially useful in fair share policies for CPU-bound workloads. Consider the following when using RCM policies to maximize CPU use:

- When considering the workload of all the partitions in a domain (the consolidated workload), the peak CPU use should not greatly exceed the average CPU use. Minimizing the gap between peak CPU load and average CPU load maximizes the CPU use for the domain.
- Oracle recommends configuring RCM CPU policies so that about 75 percent of CPU use is used for applications housed in the partitions of a domain. The remaining 25 percent should be allocated approximately as: 10 percent for operational tasks (backup, scheduled tasks, and other administration) and 15 percent for cluster failover.

Managing Heap

Develop a memory use plan that supports the requirements of the partition applications while continuing to provide enough available heap (headroom) for the domain and other system work. When evaluating heap requirements for a domain, consider the low, average, steady-state, and peak `Heap Retained` use values for each partition's representative workload.

RCM Limitations

RCM has certain limitations of which you should be aware.

- Heap resource consumption tracking and management is supported only when run with the G1 garbage collector (there is no RCM support for other JDK collectors).
- There is no support to measure and account for resource consumption metrics for activities happening in JNI/native code.
- Measurements of retained heap and CPU use are performed asynchronously and hence do not represent *current (a point-in-time)* value.
- Discrimination of heap use for objects in static fields, and singleton objects of classes loaded from system and shared classloaders are problematic and may not be accurately represented in the final accounting values. If an instance of a class loaded from system and shared classloaders is loaded by a partition, the instance's use of heap is accounted against that partition.
- Garbage collection activity is not isolated to specific domain partitions in WebLogic Server 12.2.1 with Oracle JDK 8u40.
- There is a detrimental performance effect to enabling the RCM feature due to the additional tracking and management of resource consumption in the server instance.

Configuring Resource Consumption Management: WLST Example

You can implement and monitor RCM policies using WLST.

- [RCM WLST Example: Overview](#)
- [RCM WLST Example: WLST Script](#)

RCM WLST Example: Overview

The following is an example of an RCM configuration created using WLST. In this example, a system administrator has defined:

- A Production resource manager representing the set of resource consumption management polices that the system administrator would like to establish for all production-tiered domain partitions in the domain. The Production resource manager has policies for various resources.
 - For the `FileOpen` resource type, three triggers are specified. A `Production2000` trigger ensures the partition is shut down when the number of

open file descriptors reaches 2000. A `Production1700` trigger specifies that when the number of open file descriptors exceeds 1700, the domain partition must be slowed down. A `Production1500` trigger specifies a notify action.

- For the `HeapRetained` resource type, a `Production2GB` trigger is created to ensure that when the partition's retained heap value reaches 2 GB, the partition must be shut down. A fair share value of 60 is assigned to the `Production` resource manager.
- A `Trial` resource manager that defines a similar but reduced set of policies.
- A partition named `Partition-0`.

At the completion of this script, `Partition-0` has been assigned the `Production` resource manager.

RCM WLST Example: WLST Script

The policy discussed in [RCM WLST Example: Overview](#) can be created using the WLST script in the following example:

```
startEdit()

cd('/ResourceManagement')
cd(domainName)
rm=cmo.createResourceManager('Approved')
fo=rm.createFileOpen('Approved-FO')
fo.createTrigger('Approved2000',2000,'shutdown')
fo.createTrigger('Approved1700',1700,'slow')
fo.createTrigger('Approved1500',1500,'notify')
hr=rm.createHeapRetained('Approved-HR')
hr.createTrigger('Approved2GB',2097152,'shutdown')
hr.createFairShareConstraint('FS-ApprovedShare', 60)

cd('/ResourceManagement')
cd(domainName)
rm=cmo.createResourceManager('Trial')
fo=rm.createFileOpen('Trial-FO')
fo.createTrigger('Trial1000',1000,'shutdown')
fo.createTrigger('Trial700',700,'slow')
fo.createTrigger('Trial500',500,'notify')

cd('/ResourceManagement')
cd(domainName)
rm = cmo.createResourceManager('RestartPartition')
rlp = rm.getRestartLoopProtection()
rlp.setRestartLoopProtectionEnabled(true)
rlp.setMaxRestartAllowed(3)
rlp.setMaxRestartAllowedInterval(3600)
rlp.setRestartDelay(10)
fo = rm.createFileOpen('FileOpenQuota')
fo.createTrigger('MaxFileOpenAllowed',20,'restart')

save()
activate()

startEdit()
cd('/Partitions')
cd(partition-0)
cmo.setResourceManagerRef(getMBean('/ResourceManagement/'+domainName+' /
ResourceManager/Approved'))
```

```
save()  
activate()
```

Configuring Resource Sharing: Related Tasks and Links

The referenced information provides additional information that may be useful when implementing RCM in your environment.

- The Garbage-First (G1) garbage collector is a server-style garbage collector, targeted for multiprocessor machines with large memories. It meets garbage collection pause time goals with high probability, while achieving high throughput. See <http://docs.oracle.com/javase/8/docs/technotes/guides/vm/G1.html>.
- Other WebLogic Server features that provide resource isolation and management for co-resident applications:
 - Classloader-based isolation: Classes not belonging to shared libraries of applications are isolated from each other. See [Configuring the Shared Application Classloader](#) and Understanding WebLogic Server Application Classloading in *Developing Applications for Oracle WebLogic Server*.
 - WebLogic Server Work Manager: The execution of its work is prioritized by configuring a set of scheduling guidelines. This enables the Work Manager to manage the threads allocated to an application, manage scheduling Work Manager instances to those threads, and help maintain service-level agreements. See [Configuring Partition Work Managers](#) and Using Work Managers to Optimize Scheduled Work in *Administering Server Environments for Oracle WebLogic Server*

For additional information, see Multitenancy Tuning Recommendations in *Tuning Performance of Oracle WebLogic Server*.

11

Deploying Applications

Deploying applications and modules in a multitenant environment is similar to deploying them in non-multitenant environments. To make deployments partition-specific, you can restrict them to a partition scope and override their configuration by specifying partition-specific deployment plans.

Note:

Many new Oracle WebLogic Server Multitenant (MT) features, such as resource groups, resource group templates, and application overrides, are also available in the 12.2.1 and later versions of WebLogic Server. This chapter refers to the WebLogic Server documentation where appropriate for these features.

Note that WebLogic Server Multitenant domain partitions, resource groups, resource group templates, virtual targets, and Resource Consumption Management are deprecated in WebLogic Server 12.2.1.4.0 and will be removed in the next release.

This chapter includes the following sections:

- [About Deployment Scopes](#)
- [Deploying Applications to Resource Group Templates](#)
- [Deploying Applications to Partition Resource Groups](#)
- [Deploying Applications as the Partition Administrator](#)
- [Overriding Application Configuration](#)
- [Removing an Application Override](#)
- [Using Partition-Specific Deployment Plans](#)
- [Enabling Parallel Deployment in Multitenant Environments](#)

About Deployment Scopes

Three deployment scopes make applications and libraries available at the domain level, whereas only one restricts them to a partition.

- **Global.** This is the equivalent of the domain level in a nonpartitioned environment.
- **Resource group template.** This is always at the domain level. Whether or not the application or library that you deploy to a resource group template is available at the domain level or a partition depends on the scope of the resource group that references the resource group template.
- **Resource group in a partition.** This is the only scope that is limited to a partition.

- **Resource group at the domain level.**

For more information about deployment scopes, see About Scope.

Deploying Applications to Resource Group Templates

You can deploy a common set of applications and libraries to resource group templates to simplify application configuration across a domain or domain partitions. When a resource group references a resource group template, it obtains its own runtime copies of the resources defined in the template. All the applications and libraries deployed to the resource group template then deploy to the resource group.

Resource group templates provide a convenient way to deploy the same collection of applications across multiple domain partitions. To customize application configuration, you can override the default resource group template configuration by specifying a different deployment plan for a particular application in a resource group.

For general information about deploying applications to resource group templates, including information about supported deployment operations and available deployment clients, see *Application Deployment with Resource Group Templates in Deploying Applications to Oracle WebLogic Server*.

Deploying Applications to Resource Group Templates: Main Steps

Only WebLogic Server system administrators can deploy applications and libraries to a resource group template. The main steps are as follows:

1. Specify the resource group template in which you want to deploy the application or library.
2. Specify the application or library name and directory path.
3. If using a deployment plan, then specify the deployment plan path.
4. (Optional) Specify additional deployment options. Do not specify the `targets` option, because the application uses the targets associated with the referencing resource group.

After an application is deployed to a resource group template, you can perform additional deployment operations.

5. Redeploy or update the application.

When you update an application, you can specify that WebLogic Server redeploy the original archive file or exploded directory, or you can specify that WebLogic Server deploy a new archive file in place of the original one. You can also change the deployment plan associated with the application.

6. Distribute the application.

The distribute operation copies deployment files to the targets associated with the referencing resource groups (or the default targets at the partition level if no targets exist at the resource group level) and places the application in a prepared state. You can then start the application in administration mode so you can perform final testing without opening the application to external client connections or disrupting connected clients.

7. Undeploy the application.

The undeploy operation removes the application from the resource group template and the targets associated with the referencing resource groups.

You cannot start or stop applications deployed to a resource group template, because you must perform these operations from the resource group that references the resource group template.

 **Note:**

After you perform a deployment operation for a resource group template, the operation is immediately carried out across all resource groups that reference the template. For example, if you undeploy an application from a resource group template, then the application is undeployed from all resource groups that reference that resource group template.

To deploy applications to a resource group template using Fusion Middleware Control, see *Configure resource group template deployments* in the online help.

Deploying Applications to Resource Group Templates: Examples

The following examples demonstrate how to perform deployment operations with resource group templates using WLST.

Deploying Applications to Resource Group Templates: WLST Example

The following example deploys the `sampleapp` application to the resource group template `myRGT`:

```
edit()
startEdit()
deploy(appName='sampleapp', path='path_to_application',
resourceGroupTemplate='myRGT')
activate()
```

Redeploying Applications to Resource Group Templates: WLST Example

The following example redeploys the `sampleapp` application to the resource group template `myRGT` using the deployment plan file `plan.xml`:

```
edit()
startEdit()
redeploy(appName='sampleapp', planPath='path_to_plan.xml',
resourceGroupTemplate='myRGT')
activate()
```

Undeploying Applications from Resource Group Templates: WLST Example

The following example undeploys the `sampleapp` application from the resource group template `myRGT`:

```
edit()
startEdit()
undeploy (appName='sampleapp', resourceGroupTemplate='myRGT')
activate()
```

Updating Applications in Resource Group Templates: WLST Example

The following example updates the `sampleapp` application in the resource group template `myRGT` using the deployment plan file `plan.xml`:

```
edit()
startEdit()
updateApplication(appName='sampleapp', planPath='path_to_plan.xml',
resourceGroupTemplate='myRGT')
activate()
```

Distributing Applications to Resource Group Templates: WLST Example

The following example distributes the `sampleapp` application to the resource group template `myRGT`:

```
edit()
startEdit()
distributeApplication (appPath='path_to_sampleapp', resourceGroupTemplate='myRGT')
activate()
```

Deploying Applications to Resource Group Templates: Related Tasks and Links

- [Configuring Resource Group Templates](#) for general information about resource group templates
- [Application Deployment with Resource Group Templates in *Deploying Applications to Oracle WebLogic Server*](#) for general information about deploying applications to resource group templates
- [Application Deployment in *Administering Oracle WebLogic Server with Fusion Middleware Control*](#) for online help about deploying applications using Fusion Middleware Control
- [WebLogic Server Resource Group Templates in *Administering Oracle WebLogic Server with Fusion Middleware Control*](#) for online help about configuring resource group templates using Fusion Middleware Control
- [weblogic.Deployer Command-Line Reference in *Deploying Applications to Oracle WebLogic Server*](#) for information about deploying applications using `weblogic.Deployer`
- [Deployment Commands in *WLST Command Reference for WebLogic Server*](#) for information about deploying applications using WLST
- [wldploy Ant Task Attribute Reference in *Developing Applications for Oracle WebLogic Server*](#) for information about deploying applications using the `wldploy` Ant task
- [Maven Plug-In Goals in *Developing Applications for Oracle WebLogic Server*](#) for information about deploying applications using Maven
- [Understanding the WebLogic Deployment API](#) for information about deploying applications using the JSR-88 Deployment API
- [DeploymentManagerMBean in *MBean Reference for Oracle WebLogic Server*](#) for information about deploying applications using the JMX Deployment API

Deploying Applications to Partition Resource Groups

You can deploy applications and libraries to resource groups at the domain or partition level and perform deployment operations on those applications, including start and stop. Resource groups conveniently group together Java EE applications and their resources into a distinct administrative unit and can be defined at the domain level, or be specific to a domain partition. A resource group can either contain resources directly or reference a resource group template containing the resources.

The resources and applications in a resource group are *fully qualified* in that you provide all the necessary information to start or connect to those resources. All of the applications deployed to a resource group use the targets associated with that resource group.

Partition resource groups allow each partition to contain a different set of resources and applications. You can perform the following deployment operations for applications and libraries with partition resource groups:

- Deploy
- Undeploy
- Redeploy
- Update
- Distribute
- Start
- Stop

If the operation succeeds, then the application or library deploys (or redeploys, starts, or so forth) to the targets associated with the specified partition resource group. If no targets exist for the partition resource group, then the application or library deploys to the default targets for the partition.

Note the following considerations for application deployment with partition resource groups:

- When performing deployment operations, you specify the partition name with the `partition` option. The specified partition must exist or the operation fails.
- Application names must be unique within each partition.
- For `deploy` and `distribute` operations, you must specify the partition resource group name with the `resourceGroup` option, unless one and only one resource group exists in the partition, in which case, `resourceGroup` is optional.

For other deployment operations, you do not specify the `resourceGroup` option. WebLogic Server derives the resource group from the partition name and unique application name. If the specified partition resource group does not exist, or you do not specify the `resourceGroup` option if required, then the deployment operation fails.

- The deployment operation fails if you specify the `targets` option when performing deployment operations for applications at the resource group level. You cannot target individual applications in the resource group, because applications use the targets associated with the resource group.

- You can deploy a specific library to a partition only once, because resources need to be unique across resource groups of a partition. Applications in the same partition resource group or other resource groups in the partition can reference the library if the targets match. Applications deployed to a partition cannot reference a library in another partition or another library in the domain.
- Resource groups can inherit application configuration from a resource group template. If a resource group references a resource group template, then any applications or libraries deployed to the resource group template immediately deploy to the referencing resource group. If needed, you can override the default resource group template application configuration to customize a particular application in a resource group. See [Overriding Application Configuration](#).
- Partition administrators can use WLST to perform deployment operations only on applications in their own partitions, not on global applications or applications in other partitions. The `partition` option is optional for partition administrators. If partition administrators use the `partition` option to specify a partition, and the partition does not match their associated partition, then the operation fails. For more information about deploying applications as a partition administrator using WLST, see [Configuring Security](#).

For information about deploying applications to domain resource groups, see [Application Deployment with Domain Resource Groups](#) in *Deploying Applications to Oracle WebLogic Server*.

Deploying Applications to Partition Resource Groups: Main Steps

The main steps for deploying applications and libraries to a resource group in a partition are as follows:

1. Specify the partition resource group in which you want to deploy the application or library.
2. Specify the application or library name and directory path.
3. If you are using a deployment plan, then specify the deployment plan path.
4. (Optional) Specify additional deployment options. Do not specify the `targets` option, because the application uses the targets associated with the resource group. If no targets exist for the resource group, then the application uses the default targets for the partition.

After an application is deployed to a resource group, you can perform additional deployment operations.

5. Start or stop the application.

Starting an application or module makes it available to WebLogic Server clients. Stopping an application or module makes it unavailable to WebLogic Server clients until you restart it.

6. Redeploy or update the application.

When you update an application, you can specify that WebLogic Server redeploy the original archive file or exploded directory, or you can specify that WebLogic Server deploy a new archive file to replace the original one. You can also change the deployment plan associated with the application.

7. Distribute the application.

The distribute operation copies deployment files to the targets associated with the resource group (or the default targets at the partition level if no targets exist at the resource group level) and places the application in a prepared state. You can then start the application in administration mode so you can perform final testing without opening the application to external client connections or disrupting connected clients.

8. Undeploy the application.

The undeploy operation removes the application from the resource group and the targets associated with the resource group.

To deploy applications to a partition resource group using Fusion Middleware Control, see Control domain partition application deployments in the online help.

Deploying Applications to Partition Resource Groups: Examples

The following examples demonstrate how to perform deployment operations to partition resource groups using WLST.

Deploying Applications to Partition Resource Groups: WLST Example

The following example deploys the `sampleapp` application to the resource group `myRG` in the partition `myPartition`:

```
edit()
startEdit()
deploy(appName='sampleapp', partition='myPartition', resourceGroup='myRG',
path='path_to_application')
activate()
```

Redeploying Applications to Partition Resource Groups: WLST Example

The following example redeploys the `sampleapp` application to the partition `myPartition` using the deployment plan file `plan.xml`.

You do not need to specify the partition resource group, because WebLogic Server derives the resource group from the unique application name and partition name.

```
edit()
startEdit()
redeploy(appName='sampleapp', planPath='path_to_plan.xml', partition='myPartition')
activate()
```

Undeploying Applications from Partition Resource Groups: WLST Example

The following example undeploys the `sampleapp` application from the partition `myPartition`.

You do not need to specify the partition resource group, because WebLogic Server derives the resource group from the unique application name and partition name.

```
edit()
startEdit()
undeploy (appName='sampleapp', partition='myPartition')
activate()
```

Updating Applications in Partition Resource Groups: WLST Example

The following example updates the `sampleapp` application in the partition `myPartition` using the deployment plan file `plan.xml`.

You do not need to specify the partition resource group, because WebLogic Server derives the resource group from the unique application name and partition name.

```
edit()
startEdit()
updateApplication(appName='sampleapp', planPath='path_to_plan.xml',
partition='myPartition')
activate()
```

Distributing Applications to Partition Resource Groups: WLST Example

The following example distributes the `sampleapp` application to the resource group `myRG` in the partition `myPartition`:

```
edit()
startEdit()
distributeApplication (appPath='path_to_sampleapp', resourceGroup='myRG',
partition='myPartition')
activate()
```

Starting Applications on Partition Resource Groups: WLST Example

The following example starts the `sampleapp` application on the partition `myPartition`.

You do not need to specify the partition resource group, because WebLogic Server derives the resource group from the unique application name and partition name.

```
edit()
startEdit()
startApplication (appName='sampleapp', partition='myPartition')
activate()
```

Stopping Applications on Partition Resource Groups: WLST Example

The following example stops the `sampleapp` application on the partition `myPartition`.

You do not need to specify the partition resource group, because WebLogic Server derives the resource group from the unique application name and partition name.

```
edit()
startEdit()
stopApplication (appName='sampleapp', partition='myPartition')
activate()
```

Deploying Applications to Partition Resource Groups: REST Example

For an example of a partition user in the Deployer role deploying applications using the Representational State Transfer (REST) APIs, see *Deploying Partition-Scoped Applications in [Administering Oracle WebLogic Server with RESTful Management Services](#)*.

Deploying Applications to Partition Resource Groups: Related Tasks and Links

- Configuring Domain Partitions for general information about partitions
- Configuring Resource Groups for general information about resource groups
- Application Deployment with Domain Resource Groups in *Deploying Applications to Oracle WebLogic Server* for information about deploying applications to resource groups at the domain level
- Application Deployment in *Administering Oracle WebLogic Server with Fusion Middleware Control* for online help about deploying applications using Fusion Middleware Control
- WebLogic Server Domain Partitions in *Administering Oracle WebLogic Server with Fusion Middleware Control* for online help about configuring partitions using Fusion Middleware Control
- WebLogic Server Resource Groups in *Administering Oracle WebLogic Server with Fusion Middleware Control* for online help about configuring resource groups using Fusion Middleware Control
- `weblogic.Deployer` Command-Line Reference in *Deploying Applications to Oracle WebLogic Server* for information about deploying applications using `weblogic.Deployer`
- Deployment Commands in *WLST Command Reference for WebLogic Server* for information about deploying applications using WLST
- `wldeploy` Ant Task Attribute Reference in *Developing Applications for Oracle WebLogic Server* for information about deploying applications using the `wldeploy` Ant task
- Maven Plug-In Goals in *Developing Applications for Oracle WebLogic Server* for information about deploying applications using Maven
- Understanding the WebLogic Deployment API for information about deploying applications using the JSR-88 Deployment API
- [DeploymentManagerMBean](#) in *MBean Reference for Oracle WebLogic Server* for information about deploying applications using the JMX Deployment API

Deploying Applications as the Partition Administrator

Partition administrators can perform deployment operations only on applications in their own partition, not on global applications or applications from other partitions. Likewise, they can call deployment APIs only on applications in their partition. When calling deployment APIs that return a list of applications or targets, only applications and targets in their partition will be returned.

WebLogic Server system administrators can deploy applications to partitions on behalf of partition administrators. When using the JMX API to deploy applications in a partition, you must obtain the `DeploymentManagerMBean` instance under the relevant partition (instead of using the domain-level `DeploymentManagerMBean` instance). The following examples show how system administrators and partition administrators perform deployment operations in a partition using the JMX Deployment API.

- When a system administrator connects to a global domain (for example, `t3://localhost:7001`):
 - The JMX lookup must include `DomainPartitionRuntime=partition_name` to get the `DeploymentManagerMBean` for the partition.
 - Using WLST, they must go to the partition's `DomainPartitionRuntimeMBean` to get the `DeploymentManagerMBean` for `partition_name`. For example:

```
connect('system_admin', 'gumby1234', 't3://localhost:7001')
domainRuntime()
cd('DomainPartitionRuntimes/partitionA')
cd('DeploymentManager/partitionA')
props=java.util.Properties()
props.setProperty("resourceGroup", "partitionAResourceGroup")
path="/scratch/user/foo.war"
cmo.deploy("foo", path, None, None, props)
```

- When a partition administrator connects to a domain partition (for example, `t3://localhost:7001/partitionA`):
 - The JMX lookup does not need to include `PartitionDomainRuntime=partitionA` to get the `DeploymentManagerMBean` for `partitionA`. It is added by the JMX container.
 - Using WLST, running the `domainRuntime()` command sets the `cmo` to the `DomainPartitionRuntimeMBean` for the partition.

```
connect('partitionA_admin', 'password', 't3://localhost:7001/partitionA')
domainRuntime()
cd('DeploymentManager/partitionA')
props=java.util.Properties()
props.setProperty("resourceGroup", "partitionAResourceGroup")
path="/scratch/user/foo.war"
cmo.deploy("foo", path, None, None, props)
```

When using other deployment clients, the `partition` option is optional for partition administrators. Using `weblogic.Deployer`:

```
java weblogic.Deployer -deploy -adminurl <partition_url> -username
<partition_admin_user> -password <partition_admin_password> -name <application_name>
path_to_application
```

For example:

```
java weblogic.Deployer -deploy -username partition1_admin -password password -
adminurl t3://localhost:7001/partition1 -name foo /scratch/user/foo.war
```

Overriding Application Configuration

When a resource group references a resource group template, it obtains its own runtime copy of the resources and applications defined in the resource group template. Because the resource group inherits the application configuration defined in the resource group template, you do not need to manually deploy and configure each application. Applications and libraries deployed to a resource group template immediately deploy to the referencing resource group (or undeploy, redeploy, and so forth, depending on the performed deployment operation).

If you need to customize a particular application in a resource group, then you can override the default application configuration by specifying a different deployment plan. In that deployment plan, you override the resource group template deployment

parameters with the values that you want to use for the particular application. The entire deployment plan file is overridden, not just an individual entry inside of the original deployment plan file. The application or module is then updated or redeployed using the new deployment plan for its application configuration instead of the values set in the resource group template.

 **Note:**

If you undeploy an application from a resource group template, then any application configuration overrides defined in the referencing resource group are removed.

If you redeploy an application to a resource group template, then any application configuration overrides defined in the referencing resource group from the previous deployment of that application are removed. In this scenario, you must override the application configuration again for the resource group to preserve the override values.

Whether you update or redeploy the application to use the override deployment plan depends on the type of properties that you are overriding. Use the `update` command only when overriding dynamic properties; the `update` command does not cause any disruption to the application. Use the `redeploy` command when overriding dynamic and nondynamic properties; the `redeploy` command does cause disruption to the application.

You can override application configuration for resource groups at the domain or partition level. For information about overriding application configuration for domain resource groups, see *Overriding Application Configuration in Deploying Applications to Oracle WebLogic Server*.

Overriding Application Configuration: Main Steps

The main steps for customizing a particular application in a resource group by overriding the default application configuration defined in the inherited resource group template are as follows:

1. Create a different deployment plan to use for the application that you want to customize.
2. In the new deployment plan, define the values for the attributes that you want to override for that application.
3. Specify the new override deployment plan when redeploying or updating the application.

The application now uses the override deployment plan for its configuration instead of the default template configuration.

If multiple resource groups reference the same resource group template, then ensure that you override the application configuration for all necessary resource groups.

Overriding Application Configuration: WLST Example

The following examples demonstrate:

1. Deploying the application `sampleapp` to the resource group template `myRGT` using the deployment plan file `plan.xml`.

```
edit()
startEdit()
deploy(appName='sampleapp', resourceGroupTemplate='myRGT',
path='path_to_application', planPath='path_to_plan.xml')
activate()
```

2. Updating or redeploying the application `sampleapp` to use the override deployment plan `override_plan.xml` for the referencing partition resource group in the partition `myPartition`.

You do not need to specify the partition resource group, because WebLogic Server derives the resource group from the unique application name and partition name.

Use the `update` command only when overriding dynamic attributes for an application. Use the `redeploy` command when overriding dynamic or nondynamic attributes for an application.

```
edit()
startEdit()
updateApplication(appName='sampleapp', planPath='path_to_override_plan.xml',
partition='myPartition')
activate()
```

```
edit()
startEdit()
redeploy(appName='sampleapp', planPath='path_to_override_plan.xml',
partition='myPartition')
activate()
```

Overriding Application Configuration: Related Tasks and Links

- [Overriding Application Configuration in *Deploying Applications to Oracle WebLogic Server*](#) for information about overriding application configuration at the domain level
- [Configuring Resource Overrides](#) for information about overriding resource configuration, such as JMS and JDBC
- [Override application configuration in *Administering Oracle WebLogic Server with Fusion Middleware Control*](#) for online help about overriding application configuration using Fusion Middleware Control
- [weblogic.Deployer Command-Line Reference in *Deploying Applications to Oracle WebLogic Server*](#) for information about overriding application configuration using `weblogic.Deployer`
- [Deployment Commands in *WLST Command Reference for WebLogic Server*](#) for information about overriding application configuration using WLST
- [wldploy Ant Task Attribute Reference in *Developing Applications for Oracle WebLogic Server*](#) for information about overriding application configuration using the `wldploy` Ant task
- [Maven Plug-In Goals in *Developing Applications for Oracle WebLogic Server*](#) for information about overriding application configuration using Maven
- [Understanding the WebLogic Deployment API](#) for information about overriding application configuration using the JSR-88 Deployment API

- [DeploymentManagerMBean](#) in *MBean Reference for Oracle WebLogic Server* for information about overriding application configuration using the JMX Deployment API

Removing an Application Override

If you no longer need to customize a particular application in a resource group, then you can remove an application override by removing the deployment plan containing the overridden attributes.

Then, the application configuration returns to its default configuration.

Removing an Application Override: Main Steps

The main steps for removing an application override from a resource group are as follows:

1. Select the application containing the override.
2. To remove existing overrides for dynamic attributes, update the application and specify the `removePlanOverride` option.
3. To remove existing overrides for dynamic and nondynamic attributes, redeploy the application and specify the `removePlanOverride` option.

To remove an application override using Fusion Middleware Control, see [Remove application configuration overrides](#) in the online help.

Removing an Application Override: WLST Example

The following example redeploys the application `sampleapp` to remove an existing overridden deployment plan using the `removePlanOverride` option:

```
edit()
startEdit()
redeploy(appName='sampleapp', removePlanOverride, partition='myPartition')
activate()
```

Removing an Application Override: Related Tasks and Links

- [Removing an Application Override](#) in *Deploying Applications to Oracle WebLogic Server* for information about removing an application override at the domain level
- [Configuring Resource Overrides](#) for information about overriding resource configuration, such as JMS and JDBC
- [Remove application configuration overrides](#) in *Administering Oracle WebLogic Server with Fusion Middleware Control* for online help about removing application overrides using Fusion Middleware Control
- [weblogic.Deployer Command-Line Reference](#) in *Deploying Applications to Oracle WebLogic Server* for information about removing application overrides using `weblogic.Deployer`
- [Deployment Commands](#) in *WLST Command Reference for WebLogic Server* for information about removing application overrides using WLST

- [wldploy Ant Task Attribute Reference](#) in *Developing Applications for Oracle WebLogic Server* for information about removing application overrides using the `wldploy` Ant task
- [Maven Plug-In Goals](#) in *Developing Applications for Oracle WebLogic Server* for information about removing application overrides using Maven
- [Understanding the WebLogic Deployment API](#) for information about removing application overrides using the JSR-88 Deployment API
- [DeploymentManagerMBean](#) in *MBean Reference for Oracle WebLogic Server* for information about removing application overrides using the JMX Deployment API

Using Partition-Specific Deployment Plans

You can use partition-specific deployment plans to customize an application deployed to a resource group template or resource group within a partition. Partition-specific deployment plans are especially useful in environments with different partitions that share the same set of applications, but require some customization for each partition.

You specify partition-specific deployment plans at the resource group level within a partition, not at the partition level. When WebLogic Server applies a partition-specific deployment plan to a specified application, the changes prescribed by the plan affect only the application deployment within that partition.



Note:

If you specify an application deployment plan in a resource group template, and specify a partition-specific deployment plan for the same application, then the partition-specific deployment plan overrides the resource group template plan and the partition-specific deployment plan is used.

To specify partition-specific application deployment plans, use the `redeploy` or `updateApplication` WLST commands.

Using Partition-Specific Deployment Plans: Main Steps

The main steps for using partition-specific deployment plans are as follows:

1. Create the partition-specific application deployment plan that you want to use.
2. Use the `redeploy` or `updateApplication` WLST command to redeploy or update the application using the partition-specific deployment plan.

Use the `redeploy` command when changes made to the application or module from the partition-specific deployment plan are nondynamic or dynamic. Use the `updateApplication` command only when the changes made to the application or module from the partition-specific deployment plan are dynamic.

3. Identify the scope containing the application deployment for which you want to use the partition-specific deployment plan by specifying the corresponding WLST option:

Scope	WLST option
resource group template	resourceGroupTemplate
domain resource group	none You do not need to identify the resource group name for domain resource groups. Because application names must be unique within the domain, WebLogic Server derives the resource group from the application name.
partition resource group	partition You do not need to identify a resource group name for partition resource groups. Because application names must be unique within a partition, WebLogic Server derives the resource group from the application name and partition name.

- Use the WLST options `planPath` and `appName` to identify the partition-specific deployment plan location and the name of the application deployment to modify.
- Activate your changes.

The specified application now uses the partition-specific deployment plan for its configuration.

Using Partition-Specific Deployment Plans: WLST Example

The following examples demonstrate:

- Deploying the application `sampleapp` to the resource group template `myRGT`. The partition `myPartition` references the resource group template `myRGT`.

```
edit()
startEdit()
deploy(appName='sampleapp', resourceGroupTemplate='myRGT',
path='path_to_application')
activate()
```

- Updating or redeploying the application `sampleapp` to use the partition-specific deployment plan `partition_plan.xml` in partition `myPartition`.

Use the `update` command only when changing dynamic attributes for an application. Use the `redeploy` command when changing dynamic or nondynamic attributes for an application.

```
edit()
startEdit()
updateApplication(appName='sampleapp', planPath='path_to_partition_plan.xml',
partition='myPartition')
activate()
```

```
edit()
startEdit()
redeploy(appName='sampleapp', planPath='path_to_partition_plan.xml',
partition='myPartition')
activate()
```

Using Partition-Specific Deployment Plans: Related Links and Tasks

- Configuring Resource Overrides for information about overriding resource configuration, such as JMS and JDBC

- Deployment Commands in *WLST Command Reference for WebLogic Server* for information about removing application overrides using WLST

Enabling Parallel Deployment in Multitenant Environments

Parallel deployment improves performance for use cases involving the deployment of multiple applications, the deployment of a single application with multiple modules, or the deployment of one or more applications across multiple partitions.

When an application is deployed to multiple partitions, WebLogic Server views each instance of the application as an independent application. Without parallel deployment enabled, each instance of the application is deployed to each partition sequentially.

To improve performance, you can enable parallel deployment so that the instances of the application deploy in parallel to each other:

- To enable parallel deployment of applications across partitions, configure the `ParallelDeployApplications` attribute of the `PartitionMBean`.

With this attribute enabled, applications with the same deployment order will deploy in parallel to each other. The partition-level setting overrides the domain-level setting for applications in a partition.

The overall deployment order of resources and applications remains unchanged, so deployment types that precede applications in the standard deployment order (for example, JDBC system resources and Java EE libraries) are guaranteed to be fully deployed before applications are deployed.

- To enable parallel deployment of modules within applications deployed in the partition, configure the `ParallelDeployApplicationModules` attribute of the `PartitionMBean`.
- To enable parallel deployment of modules on a per-application basis, configure the `ParallelDeployModules` attribute of the `AppDeploymentMBean`. Setting this attribute overrides the domain and partition value for an individual application.

For domains created with WebLogic Server 12.2.1 or later, the `ParallelDeployApplications` and `ParallelDeployApplicationModules` attributes are enabled by default. For domains created prior to WebLogic Server 12.2.1, these attributes are disabled by default.

Note:

Only applications with no cross-dependencies should deploy in parallel. If an application depends on other applications, then this application needs to have a higher dependency order than the applications it depends on. Otherwise, parallel deployment may cause dependency failures when the dependent application attempts to deploy prior to the activation of applications on which it depends. Similarly, enable parallel deployment of modules only for applications that contain modules with no cross-dependencies.

For more information about parallel deployment, see *Enabling Parallel Deployment for Applications and Modules in Deploying Applications to Oracle WebLogic Server*.

12

Configuring the Shared Application Classloader

When running multiple instances of the same application in different partitions, you may want to optimize classloading memory across these instances. WebLogic Server MT includes a deployment-descriptor-based classloading feature for this purpose.

Note:

WebLogic Server Multitenant domain partitions, resource groups, resource group templates, virtual targets, and Resource Consumption Management are deprecated in WebLogic Server 12.2.1.4.0 and will be removed in the next release.

The classloader feature is intended for use only by application developers. This chapter assumes that you are familiar with WebLogic Server application classloading, as described in *Understanding WebLogic Server Application Classloading in Developing Applications for Oracle WebLogic Server*. Read that chapter first if you are not familiar with WebLogic Server application classloading.

This chapter includes the following sections:

- [Configuring the Shared Application Classloader: Overview](#)
- [Configuring the Shared Application Classloader: Main Steps](#)
- [Configuring the Shared Application Classloader: Related Tasks and Links](#)

Configuring the Shared Application Classloader: Overview

WebLogic Server MT includes two new types of classloaders: partition classloaders and shared application classloaders.

- **Partition classloader:** A partition classloader is created on your behalf for each partition. This classloader loads classes that must be visible to the entire partition and only that partition, and not to the rest of the domain. (This includes **global connectors**.)

Note:

Global connectors are a certain type of connector modules that are configured to be visible and available for all the applications in the server. These modules may be deployed standalone or may be deployed as a module of an Enterprise Archive (EAR) file.

- Shared application classloader: The shared application classloader is relevant only in the Software-as-a-Service (SaaS) use case when an application is deployed to a resource group template. This classloader loads classes that should be visible to all the instances of an application across partitions, but not to other applications. Applications directly deployed to the domain (not to a partition) are not affected, even if partitions are configured.

WebLogic Server deployment descriptors support a new `shareable` configuration to identify JAR files and classes under the `APP-INF/classes` directory within the application packaging. These classes should be loaded from the shared application classloader.

The `shareable` configuration is supported only in the `weblogic-application.xml` file of enterprise applications, and only in production mode.

A new partition classloader is created when a partition is created and the first application for the partition is deployed to a resource group. An existing partition classloader is removed when a partition is removed and the last application for the partition is undeployed from a resource group.

A new shared application classloader may be created when an application (with sharable JAR files configured) is deployed to a resource group template and the resource group template is referenced by a resource group in a partition.

An existing shared application classloader is removed when the relevant application is undeployed from the resource group template. When you undeploy an application from a resource group template, it is immediately undeployed from any resource groups that reference that template.

Note:

You cannot assume that classloading will be shared. Sharing may or may not be possible. For example, if filtering is set up such that different configurations for each partition are obtained, then sharing is not possible. For information about filtering classloaders, see *Using a Filtering ClassLoader* in *Developing Applications for Oracle WebLogic Server*.

Therefore, you can indicate which JAR files are shareable, but cannot assume that the classes from these JAR files are loaded only once and shared.

Configuring the Shared Application Classloader: Main Steps

To configure a shared application classloader, application developers, using an XML editor, can edit the `weblogic-application.xml` file and add one or more `shareable` elements inside a `<class-loading>` element when creating the application.

Consider the following example:

```
<class-loading>
  <shareable dir="APP-INF-LIB">
    <include>coupon-generator.jar</include>
    <include>group-discounts.jar</include>
  </shareable>
  <shareable dir="LIB-DIR">
```



```

    <exclude>program-guide.jar</exclude>
  </shareable>
</class-loading>

```

The `dir` attribute identifies the directory where subsequent patterns apply. The only supported values for `dir` are as follows:

- `APP-INF-LIB` identifies the WebLogic Server-style `APP-INF/lib` directory.
- `LIB-DIR` identifies the Java EE-style library directory.
- `APP-INF-CLASSES` identifies the WebLogic Server-style `APP-INF/classes` directory.

For the `APP-INF-LIB` and `LIB-DIR` directories, the value that you set in `<include></include>` elements identifies JAR files that you believe can be shared without any problems. The classes in these JAR files cannot be dependent on:

- Classes in JAR files that are not considered shareable.
- Partition-scoped classes such as global connectors.

Similarly, for the `APP-INF-LIB` and `LIB-DIR` directories, the `<exclude></exclude>` elements helps identify JAR files that must not be shared.

The following configuration rules apply:

- If `APP-INF-LIB` and `LIB-DIR` are identified without an include or exclude configuration, then all the JAR files of that directory are considered sharable.
- If `APP-INF-LIB` and `LIB-DIR` are identified with an include configuration, then only the JAR files identified with the include pattern are considered sharable.
- If `APP-INF-LIB` and `LIB-DIR` are identified with an exclude configuration, then all the JAR files of the directory except the JAR file identified by the exclude pattern are considered sharable.

APP-INF-CLASSES

For the `APP-INF-CLASSES` directory, include and exclude configurations are ignored, even if present. If you identify `APP-INF-CLASSES` as sharable, then all the classes are declared sharable.

The `APP-INF-CLASSES` directory is a code source location and does not require include and exclude elements. The shared application classloader implementation does not allow including or excluding elements within code source locations.

The `APP-INF-CLASSES` directory is valid only for the packaging unit the descriptor is in, whether it is an application or library. You must declare it separately for the application and libraries.

Configuring the Shared Application Classloader: Related Tasks and Links

See the referenced information to learn about filtering classloaders.

Using a Filtering ClassLoader in *Developing Applications for Oracle WebLogic Server*.

13

Configuring Oracle Coherence

Oracle Coherence applications can take full advantage of the density and operational efficiencies that are provided by WebLogic Server MT. All tenant instrumentation is automatically provided without any changes required to the application.

Note:

WebLogic Server Multitenant domain partitions, resource groups, resource group templates, virtual targets, and Resource Consumption Management are deprecated in WebLogic Server 12.2.1.4.0 and will be removed in the next release.

This chapter includes the following sections:

- [Configuring Oracle Coherence: Overview](#)
- [Deploying Oracle Coherence Applications in Multitenant Domains](#)
- [Overriding Cache Configuration Properties](#)
- [Enabling Cache Sharing Across Partitions](#)
- [Securing Oracle Coherence Applications in Multitenant Domains](#)
- [Configuring Oracle Coherence: Related Tasks and Links](#)

Configuring Oracle Coherence: Overview

Multitenant domains use a single shared Oracle Coherence cluster that is available across all domain partitions. Application caches are isolated, and can also be shared, at the domain partition level. Common deployment tooling is used to deploy and manage Oracle Coherence Grid ARchive (GAR) modules and common partition lifecycle operations are used to manage the life cycle of Oracle Coherence services.

Understanding Oracle Coherence Setup in a Multitenant Domain

Oracle Coherence setup in a multitenant domain requires the use of managed Oracle Coherence servers. You must first create an Oracle Coherence cluster for the domain and configure all managed Oracle Coherence servers to be part of the cluster. Managed Oracle Coherence servers are typically set up in tiers using WebLogic Server clusters, which allows Oracle Coherence to easily scale up or scale down as required. For details, see *Administering Clusters for Oracle WebLogic Server*. In addition, Oracle Coherence applications must be packaged as GAR modules to be deployed to a multitenant domain. For more information about packaging an Oracle Coherence application as a GAR module, see *Developing Oracle Coherence Applications for Oracle WebLogic Server*.

Understanding Cache Isolation in a Multitenant Domain

Cache isolation in a multitenant domain allows a single GAR module to be deployed across multiple domain partitions and used by all tenants. Isolation is provided at the domain partition level and is transparent to the application. When a GAR module is deployed to a domain partition, a new cache service instance is created and is isolated to that domain partition. The GAR module generates the appropriate `tenant-id` and `meta-data` when starting the Oracle Coherence services. Some cache configuration properties may be overridden with different values for each domain partition.

Understanding Cache Sharing

Cache sharing allows a single cache instance to be used by multiple domain partitions. Each tenant has access to the same cache. Cache sharing is enabled at the partition level and must be set for each domain partition that accesses the cache. Cache sharing is transparent to the application.

Cache sharing provides the greatest amount of cache reuse and the most efficient use of resources. However, cache sharing should be used only when cache data is not specific to any one tenant.

Deploying Oracle Coherence Applications in Multitenant Domains

Oracle Coherence applications are deployed using the common deployment tools such as Fusion Middleware Control and WLST.

For more information about deploying applications in a multitenant environment, see [Deploying Applications](#).

When deploying Oracle Coherence applications, use the Scope field to specify the domain partition to which the application is deployed. Oracle Coherence applications can be deployed to the following:

- Global scope: The global scope is equivalent to a non-multitenant deployment and does not impose any cache isolation.
- Resource group template: Applications are deployed to the domain partitions that are configured to use the specified resource group template. Cache instances are isolated to each domain partition.
- Resource group for a partition: Applications are deployed to the partitions that are configured to use the specified resource group. Cache instances are isolated to each domain partition.

Overriding Cache Configuration Properties

Cache configuration properties can be specified for each domain partition. Partition-specific values are inserted into the cache configuration when a GAR module is initialized. For example, a cache in one partition may be configured so that entries expire after 5 minutes and for another domain partition the cache may be configured so that entries expire after 10 minutes. In both cases, the same cache configuration file is deployed in the GAR module.

The main steps for overriding cache configuration properties are as follows:

1. Select a partition to edit its partition settings.
2. Update Oracle Coherence configuration settings by adding the property names and values to be overridden.

To override cache configuration properties using Fusion Middleware Control, see [Configure domain partition Oracle Coherence caches in the online help](#).

Using WLST

To override cache configuration properties, connect to the Weblogic Server Domain MBean server and execute the `createCoherencePartitionCacheProperty` operation to specify a property and the `setValue` operation to enter the property value. For example:

```
cd('/')
cd('/Partitions/MyPartition-1')
cmo.createCoherencePartitionCacheConfig('MyConfig')
cd('CoherencePartitionCacheConfigs/MyConfig')
cmo.createCoherencePartitionCacheProperty('expiry-delay')
cd('CoherencePartitionCacheProperties/expiry-delay')
cmo.setValue('3600s')
```

Cache configuration properties can also be deleted using the `destroyCoherencePartitionCacheConfig` operation. For example:

```
cd('/')
cd('/Partitions/MyPartition-1')
cmo.createCoherencePartitionCacheConfig('MyConfig')
cd('CoherencePartitionCacheConfigs/MyConfig/CoherencePartitionCacheProperties/
  expiry-delay')
property = cmo
cd('..')
cmo.destroyCoherencePartitionCacheProperty(property)
```

Enabling Cache Sharing Across Partitions

Cache sharing must be explicitly set for each partition where the cache is deployed. Enabling cache sharing on a cache in one partition does not automatically share the cache for all partitions.

Note that the same cache scheme cannot be used for both shared and nonshared caches in a multitenant domain. However, a cache scheme can use scheme references to reuse the same scheme definition. For example:

```
<caching-scheme-mapping>
  <cache-mapping>
    <cache-name>example</cache-name>
    <scheme-name>ExamplesPartitioned-NonShared</scheme-name>
  </cache-mapping>
  <cache-mapping>
    <cache-name>example-shared</cache-name>
    <scheme-name>ExamplesPartitioned-Shared</scheme-name>
  </cache-mapping>
</caching-scheme-mapping>
<caching-schemes>
  <distributed-scheme>
    <scheme-name>ExamplesPartitioned-NonShared</scheme-name>
    <scheme-ref>ExamplesPartitionedScheme</scheme-ref>
  </distributed-scheme>
```

```

<distributed-scheme>
  <scheme-name>ExamplesPartitioned-Shared</scheme-name>
  <scheme-ref>ExamplesPartitionedScheme</scheme-ref>
</distributed-scheme>
<distributed-scheme>
  <scheme-name>ExamplesPartitionedScheme</scheme-name>
  <service-name>ExamplesPartitionedCache</service-name>
  <backing-map-scheme>
    <local-scheme>
      <high-units>32M</high-units>
      <unit-calculator>binary</unit-calculator>
    </local-scheme>
  </backing-map-scheme>
  <autostart>true</autostart>
</distributed-scheme>
</caching-schemes>

```

The main steps for enabling cache sharing are as follows:

1. Select a partition to edit its partition settings.
2. Add an Oracle Coherence cache to the partition.
3. Specify that the cache is shared.
4. Repeat the process for each partition that shares the cache. Configuration values must be the same for each partition.

To enable cache sharing using Fusion Middleware Control, see [Configure domain partition Oracle Coherence caches](#) in the online help.

Using WLST

To enable cache sharing across partitions, connect to the WebLogic Server Domain MBean server and execute the `setShared` operation with a `true` parameter. For example:

```

cd('/')
cd('/Partitions/MyPartition-1')
cmo.createCoherencePartitionCacheConfig('MyConfig')
cd('CoherencePartitionCacheConfigs/MyConfig')
cmo.setShared(true)

cd('/')
cd('/Partitions/MyPartition-2')
cmo.createCoherencePartitionCacheConfig('MyConfig')
cd('CoherencePartitionCacheConfigs/MyConfig')
cmo.setShared(true)

```

Securing Oracle Coherence Applications in Multitenant Domains

Oracle Coherence applications use the authorization features that are available with WebLogic Server. Authorization roles and policies are configured for caches and services and should be set for each partition.

For more information about security in multitenant domains, see [Configuring Security](#). For more information about Oracle Coherence security in WebLogic Server, see [Securing Oracle Coherence](#).

 **Note:**

Shared caches are owned by the global partition and authorization is based on the global realm.

Configuring Oracle Coherence: Related Tasks and Links

For a detailed example of using Oracle Coherence with multitenancy, see the Managed Oracle Coherence Servers - multitenant example that is included in the WebLogic Server Code Examples. For more information about creating Oracle Coherence clusters, packaging Oracle Coherence applications, and securing Oracle Coherence in Oracle WebLogic Server, see the referenced information.

- *Administering Clusters for Oracle WebLogic Server.*
- *Developing Oracle Coherence Applications for Oracle WebLogic Server.*
- *Securing Oracle Coherence*

14

Configuring JDBC

Learn how data source resource definitions are supported in Oracle WebLogic Server Multitenant (MT).

Note:

WebLogic Server Multitenant domain partitions, resource groups, resource group templates, virtual targets, and Resource Consumption Management are deprecated in WebLogic Server 12.2.1.4.0 and will be removed in the next release.

This chapter includes the following sections:

- [Configuring JDBC: Prerequisites](#)
- [About Supported Data Source Types](#)
- [Configuring JDBC Data Sources: Considerations](#)
- [Configuring JDBC Data Sources: WLST Example](#)
- [Configuring JDBC Data Sources: Administration Console Example](#)
- [Configuring JDBC Data Sources: Fusion Middleware Control Example](#)
- [Configuring JDBC Data Sources: REST Example](#)
- [Configuring JDBC Data Sources: Related Tasks and Links](#)

Configuring JDBC: Prerequisites

Prior to configuring Java Database Connectivity (JDBC) in a multitenant environment, you must complete certain prerequisite steps.

You must have already created:

- A virtual target. See [Configuring Virtual Targets](#).
- A domain partition. See [Configuring Domain Partitions](#).
- A security realm that is specific to the partition, if necessary. See [Configuring Security](#).
- A resource group template, if necessary. See [Configuring Resource Group Templates](#).
- A resource group. See [Configuring Resource Groups](#).

This chapter assumes familiarity with existing WebLogic Server data sources. See *Administering JDBC Data Sources for Oracle WebLogic Server*.

About Supported Data Source Types

When working in a non-multitenant WebLogic Server environment, a data source may be defined as a system resource or deployed at the domain level. When using WebLogic Server MT in a multitenant environment, a data source may also be defined in one of several scopes.

- As part of a resource group that is created at the domain level. The referenced data source must be qualified with the appropriate database credentials, URL, driver properties, and attributes for the domain-level scope. A data source referenced by a resource group at the domain level is a global system resource. A data source module descriptor may be referenced only by a single domain-level resource group.
- As part of a resource group that is created at the partition level. The referenced data source must be qualified with the appropriate database credentials, URL, driver properties, and attributes for the partition. A data source descriptor can be referenced only by a single resource group in a partition.
- As part of a resource group template that is created at the domain level. Such definitions are considered a template data source definition that is fully qualified by individual partitions that reference the resource group template.
- As part of a resource group at the partition level that is based on a resource group template. Because template resource descriptors referenced by a resource group template may be incomplete, by lacking or having an incorrect URL, credentials, or both, the resources defined by each resource group need to have the appropriate overrides specified for deployment to the partition runtime.

You can use a `JDBCSystemResourceOverrideMBean` to override the `user`, `password`, and `URL` attributes for a resource group based on a resource group template that is defined at the partition level. These three attributes represent data source settings that would typically be qualified on a per-partition basis for Software-as-a-Service (SaaS) use cases where most template data source settings are common across partitions. If you use override MBeans, then you must define a separate override MBean for each data source referenced by the resource group. Configuration changes to these attributes that are made at runtime (post data source deployment) require that the partition data source be restarted for the changes to take effect.

A resource deployment plan may be specified for a resource group that allows overriding arbitrary attributes of any data source descriptor referenced by the resource group. The single deployment plan is defined in an external file whose path is specified in the partition configuration and applies to all data sources in the partition. The resource deployment plan is processed before applying override attributes to the resource group-referenced data source. If a given data source attribute is specified in both a resource deployment plan and an override MBean, then the override MBean value takes precedence. See [Configuring Resource Overrides](#).

 **Note:**

For any Java EE applications that use JDBC directly, you must update the application to use data sources instead if you want to use JDBC overrides.

- As part of a deployed object at the partition-level. This includes a packaged data source in an EAR or WAR file, in a standalone data source module, or a Java EE data source definition defined in a Java annotation or a descriptor file in an application. Each of these objects can be deployed to a partition. Packaged and standalone data sources can be updated using application deployment plans, the same as the corresponding nonpartitioned data sources. There is no override mechanism for Java EE data source definitions.

To summarize, the data source component supports the following data source deployment types.

Data Source Deployment Type	Parameter Override Support
Domain-level system resource, optionally scoped in a resource group	No override support; change the data source directly.
Resource group template system resource	Change the data source directly or override in the resource group derived from the resource group template.
Partition-level system resource in a resource group	No override support; change the data source directly.
Partition-level system resource in a resource group based on a resource group template	Use a JDBC system resource override or a resource deployment plan.
Application scoped/packaged data source deployed to a domain or partition	Use an application deployment plan.
Standalone data source module deployed to a domain or partition	Use an application deployment plan.
Data source definitions (Java EE 6) deployed to a domain or partition	No override support.

The domain structure related to data sources is as follows:

- Domain:
 - Data source with global scope
 - Domain-level resource group with data source with global scope
 - Domain-level resource group template with data source
 - Partition:
 - * Partition-level resource group with data source
 - * Partition-level resource group based on resource group template
 - * Partition-level JDBC system resource override
 - * Partition-level resource deployment plan
 - * Object deployed at the partition level

The resource group template references system resource definitions such as data source module descriptors. The partition defines one or more resource groups that may reference a system resource definition directly, a resource group template, or both.

Configuring JDBC Data Sources: Considerations

When configuring data sources in a multitenant environment, you should take into consideration several topics, including data source scope, runtime monitoring, security roles and policy definitions, the scope of XA transactions, and more.

- [Data Source Scope](#)
- [Runtime Monitoring](#)
- [Security](#)
- [Transactions Scope](#)
- [Partition Life Cycle](#)
- [Diagnostic Image Source](#)
- [Logging](#)
- [JNDI Bindings](#)
- [Deprecated Drivers Not Supported](#)

Data Source Scope

Creating a data source that is scoped to a domain-level resource group or in a partition is similar to creating a domain-level system resource. The only additional step is to specify the scope. In the Oracle WebLogic Server Administration Console and Oracle Enterprise Manager Fusion Middleware Control (FMWC), there is a menu in the first step of the creation process that lists the available scopes in which to create a data source. Using WebLogic Scripting Tool (WLST), it is necessary to create the data source using the `createJDBCSystemResource` method on the parent MBean (the MBean for the domain, resource group, or resource group template).

When editing a data source, the data source must reside in the proper scope. There are no additional changes to updating a data source.

Runtime Monitoring

The existing data source runtime MBeans are supported for partition-scoped data source deployments and are accessible to JMX-based management clients. The runtime MBean type depends on the data source type deployed. For example, a generic data source is represented by a `JDBCDataSourceRuntimeMBean` type.

The `DataSourceRuntimeMBean` created for a partition-scoped data source deployment is parented by the `JDBCPartitionRuntimeMBean` of the partition and provides statistics and runtime operations that are specific to the partition's data source instance.

Partition-scoped data source runtime MBeans have a name attribute similar to the following:

```
com.bea:ServerRuntime=myserver,Name=ds,Type=JDBCDataSourceRuntime,  
JDBCPartitionRuntime=Partition1,PartitionRuntime=Partition1
```

The parent attribute is set to:

```
com.bea:ServerRuntime=myserver,Name=Partition1,Type=JDBCPartitionRuntime
```

Note that in the prior example, the partition-scoped data source name is `ds`, the partition name is `Partition1` and the server name is `myserver`.

The data source runtime MBeans reside in the runtime MBean hierarchy under `serverRuntime/PartitionRuntimes/partition/JDBCPartitionRuntime`. Whereas the runtime MBeans for nonpartition-scoped data source deployments reside under `serverRuntime/JDBCServiceRuntime`.

MBeans defined at the domain level are not visible to partition users. If a nondynamic parameter is updated in a partition, the change does not take effect immediately but occurs the next time that the data source in the partition is restarted or the partition itself is restarted. It is not necessary to restart the server.

Security

Security roles and policy definitions related to partition data source configuration is the responsibility of the system administrator. See [Configuring Security](#). It is optional to configure a security realm that is specific to the partition. Otherwise, configuring security for a partition-scoped data source is similar to a global data source. For more information about data source security, see [Security for WebLogic Server MBeans and Understanding Data Source Security](#).

Data source authorization verification is supported for partition-scoped deployments using the partition-specific security realm where applicable.

There is support for defining credential mapper entries for use with data source security features for partition-scoped deployments. Data source credential mapper entries are used with data source security options: identity-based pooling, Set Client Identifier, and proxy authentication. Credential mapper entries should be created in the WebLogic Server Administration Console (they are not supported in Fusion Middleware Control).

Transactions Scope

The scope of XA transactions are at the domain level and span access to partition-level resources. Partition-scoped resource names are qualified with the partition name so that resources are unique to the transaction manager, and they are managed independently.

For more information about transaction configuration and restrictions, see [Configuring Transactions](#).

Partition Life Cycle

A data source that is associated with a partition is started during a partition start and shut down during a partition shutdown or forced shutdown. Partition start and shutdown can be performed by the WebLogic Server system administrator and operator, the partition administrator, and the partition operator.

Diagnostic Image Source

You can limit the scope of a diagnostic image to a partition. For diagnostic image capture requests that specify a partition, the data source output is restricted to that of data sources that are scoped to the target partition. See [Configuring Partition-Scoped Diagnostic Image Capture](#).

Logging

Partition-scoped data source log messages are qualified with the partition ID and name when the domain log format is not configured for backward compatibility with pre-12.2.1 logging. For more information about logging, see [Monitoring and Debugging Partitions](#).

JNDI Bindings

A data source is bound into the appropriate JNDI context at deployment time under the name or names specified in the data source module descriptor.

Data Source Scope	JNDI Namespace
Application	Application
Partition resource group	Partition
Domain resource group	Global

Note that a data source instance obtained from a JNDI namespace preserves its scope even if accessed under a different partition context.

Cross-partition access occurs when an application running in one partition accesses a resource in another partition. This also applies to partition-level access to the domain level and the reverse. Cross-partition data source access occurs when an application running in one partition performs a JNDI lookup of a data source defined in another partition by using either the partition URL, or the `domain:` or `partition:` syntax. In WebLogic Server 12.2.1 or later, this causes a warning to be generated in the log so that applications know that a change is required.

Deprecated Drivers Not Supported

The three WebLogic Server driver types that are registered under the JDBC URL prefix `jdbc:weblogic:pool`, `jdbc:weblogic:rmi`, and `jdbc:weblogic:jts` are deprecated and are not supported with partition-scoped data sources. Existing JDBC client logic that specifies the `connectionPoolID` property with the data source name, does not fail to find the data source in a partition. Applications that use the pool, JTS, or RMI drivers must modify their configurations to specify data source JNDI names instead to work in a partition configuration.

Related to this, an EJB RDBMS Bean descriptor cannot use `<pool-name>` to access a partition-scoped data source in a JDBC session. Using a JNDI name is supported in this situation.

Configuring JDBC Data Sources: WLST Example

This sample WLST program shows how to configure a WebLogic Server data source in an multitenant environment.

It creates two virtual targets, one for the domain and one for the partition. It creates one partition. It then creates a resource group, a resource group template, and a resource group based on the resource group template in the partition.

It then creates the same data source in four scopes:

- Resource group template
- Resource group using the resource group template and a system override
- Resource group
- Domain

Last, it starts the partition.

```
import sys, socket
import os
hostname = socket.gethostname()
partition='partition1'
connect("weblogic","password","t3://"+hostname+":7001")
edit()
startEdit()
serverBean = getMBean('/Servers/myserver')
realmBean=cmo.getSecurityConfiguration().getDefaultRealm()
pB = cmo.lookupPartition(partition)
if pB != None:
    print "[ERROR] Partition with Name '%s' already exists" % partition
    cancelEdit('y')
    exit('y')
host='%s.us.example.com' %hostname
def createVirtualTarget(name, host, domain, target, prefix):
    print "Creating virtual target " + name
    domain=getMBean('/')
    vt=domain.createVirtualTarget(name)
    vt.addTarget(target)
    vt.setHostNames(jarray.array([String(host)],String))
    vt.setUriPrefix(prefix)
    return vt
def createJDBCSystemResource(owner, resourceName):
    systemResource=owner.createJDBCSystemResource(resourceName)
    systemResource.setName(resourceName)
    jdbcResource=systemResource.getJDBCResource()
    jdbcResource.setName(resourceName)
    driverParams=jdbcResource.getJDBCDriverParams()
    driverParams.setDriverName('oracle.jdbc.OracleDriver')
    driverParams.setUrl('jdbc:oracle:thin:@dbhost:1521/otrade')
    # driverParams.setDriverName('org.apache.derby.jdbc.EmbeddedDriver')
    # driverParams.setUrl('jdbc:derby:memory:mydb\;create=true')
    properties = driverParams.getProperties()
    properties.createProperty('user', 'dbuser')
    driverParams.setPassword('password')
    jdbcDataSourceParams=jdbcResource.getJDBCDataSourceParams()
    jdbcDataSourceParams.addJNDIName(resourceName)
    jdbcDataSourceParams.setGlobalTransactionsProtocol('None')
    return systemResource
```

```
def createJDBCSystemResourceOverride(partition, dsname, url, user, password):
    oMBean=partition.createJDBCSystemResourceOverride(dsname)
    oMBean.setURL(url)
    oMBean.setUser(user)
    oMBean.setPassword(password)
    vtname='partition1'
    vtBean=createVirtualTarget(vtname+'-vtarget', host, getMBean('/'),
        getMBean('/Servers/myserver'), '/' + vtname)
    vtname='domain'
    vtBean1=createVirtualTarget(vtname+'-vtarget', host, getMBean('/'),
        getMBean('/Servers/myserver'), '/' + vtname)
    print 'Creating partition partition1'
    domain = getMBean("/")
    partitionLMBean=domain.createPartition('partition1')
    partitionLMBean.addDefaultTarget(vtBean)
    partitionLMBean.setRealm(realmBean)
    partitionLMBean.addAvailableTarget(vtBean)
    print 'Creating resource group template rgt'
    rgtBean=cmo.createResourceGroupTemplate('rgt')
    print 'Creating resource group partition1-rg in partition1'
    partitionrgMBean=partitionLMBean.createResourceGroup('partition1-rg')
    partitionrgMBean.addTarget(vtBean)
    print 'Creating resource group partition1-rg-with-template in partition1'
    partitionlrgitMBean=partitionLMBean.createResourceGroup('partition1-rg-with-
        template')
    partitionlrgitMBean.setResourceGroupTemplate(getMBean('/ResourceGroupTemplates/rgt'))
    print 'Creating domain resource group global-rg'
    cd('/')
    cmo.createResourceGroup('global-rg')
    cd("/ResourceGroups/global-rg")
    cmo.addTarget(vtBean1)
    save()
    activate()
    # Create 4 data sources
    startEdit()
    print "Creating data source ds-in-template in rgt"
    createJDBCSystemResource(owner=rgtBean, resourceName='ds-in-template')
    activate()
    startEdit()
    print "Creating data source ds-using-template in rg using rgt"
    createJDBCSystemResource(owner=partitionlrgitMBean, resourceName='ds-using-template')
    activate()
    startEdit()
    print "Creating override for data source ds-in-template"
    createJDBCSystemResourceOverride(partitionLMBean, 'ds-in-template',
        jdbc:oracle:thin:@dbhost:1521/otrade2', 'scott', 'tiger',1,10,50)
    activate()
    startEdit()
    print "Creating data source ds in partition1-rg"
    createJDBCSystemResource(owner=partitionrgMBean, resourceName='ds')
    activate()
    # You cannot create a data source directly in a partition.
    # createJDBCSystemResource(owner=partitionLMBean, resourceName='ds')
    startEdit()
    print "Creating data source ds in domain"
    createJDBCSystemResource(owner=domain, resourceName='ds')
    cd('/SystemResources/' + "ds" )
    set('Targets',jarray.array([ObjectName('com.bea:Name=' + 'AdminServer' +
        ',Type=Server')], ObjectName))
    save()
    activate()
```

```

startPartitionWait(partition1MBean)
def createJDBCSystemResourceOverride(partition, dsname, url, user, password,
initialCapacity, minCapacity, maxCapacity):
oMBean=partition.createJDBCSystemResourceOverride(dsname)
oMBean.setDataSourceName(dsname)
oMBean.setURL(url)
oMBean.setUser(user)
oMBean.setPassword(password)
oMBean.setInitialCapacity(initialCapacity)
oMBean.setMinCapacity(minCapacity)
oMBean.setMaxCapacity(maxCapacity)

```

Note that the `createJDBCSystemResource` method is unusual in that it does not have many different parameters to create the data source. Instead, the same parameters are used each time.

The following sample WLST output shows a global system resource data source deployment under the server runtime and the three partition-scoped data source deployments under the partition runtime.

```

> ls('JDBCSystemResources')
dr-- ds
> ls('Partitions')
drw- partition1
> serverRuntime()
> ls('JDBCServiceRuntime/AdminServer/JDBCDataSourceRuntimeMBeans')
dr-- ds
> ls(
'PartitionRuntimes/partition1/JDBCPartitionRuntime/partition1/
JDBCDataSourceRuntimeMBeans')
dr-- ds
dr-- ds-in-template
dr-- ds-using-template

```

Configuring JDBC Data Sources: Administration Console Example

You can also configure JDBC data sources in a multitenant environment using the WebLogic Server Administration Console. The following sections specify the steps to take using the WebLogic Server Administration Console to reproduce the [Configuring JDBC Data Sources: WLST Example](#).

Note:

When you create a data source in a resource group template, values, such as URL and user, may be overridden in the resource group. Since this is an expected use case, it is handled in a special manner by the Administration Console. If you do not fill in a value, it will be filled in with a placeholder value based on the attribute name.

Configuring JDBC Data Sources

To configure JDBC data sources:

1. From the WebLogic Server Administration Console home page, select **Data Sources** to display a summary table which lists all the system resource data sources that are configured in all scopes (domain, resource group templates, and resource groups). The table displays the scope and partition name where applicable.
2. Select the name of a data source.
3. Select **Configuration > General** to display the configuration attributes unaffected by any existing JDBC system resource overrides. The overrides do not appear at this level.
4. Select **Security > Credential Mappings** and then click **New**. Enter the WebLogic Server User, Remote User, and Remote Password values for credential mappings associated with the data source.
5. On the data sources summary page, click **New** and select **Generic Data Source**. From the **Scope** menu, you can specify a global data source or whether to create the data source in an existing resource group template or resource group (the names of all the available scopes appears in the menu).
6. From the WebLogic Server Administration Console home page, select **Domain Partitions** to display the domain partitions summary table that lists all the configured partitions. The information for each partition includes the nested resource groups, the default targets, and the state.
7. To configure a specific partition, click its name, then select **Resource Overrides > JDBC System** to display a summary table that lists the existing JDBC overrides.
8. To create a new JDBC system resource override, click **New**. The **Data Source** menu lists the available data sources for creating the override. The WebLogic Server Administration Console lists the data sources from all the resource groups in the partition, however, only resource groups derived from resource group templates would need an override. Nonderived resource groups can be updated directly; overriding them is not recommended. For each existing override, the table displays the name of the override, the data source, and the URL. Clicking the override name displays a configuration page where you can update any of the override values (URL, user, or password).

Monitoring JDBC Data Sources

To monitor data sources at various levels:

1. On the domain level, from the Home page, select **Data Sources > Monitoring** to view all running data sources in all scopes.
2. On the partition level, select **Domain Partitions** and click a specific partition name.
3. Select **Resource Groups**, click a specific resource group name, then select **Services > JDBC** to view the data sources defined in this scope.

Configuring JDBC Data Source Diagnostics

To produce a partition-scoped diagnostics image:

1. From the home page, select **Diagnostics Images**.
2. Click the plus sign (+) for the desired server to expand the list.

3. Select the radio button for a specific partition name and then click **Capture Image**.

The Diagnostic Image Properties page specifies the Destination Directory location for the image .zip file.

4. Open the .zip file to view a JDBC.txt file that might look like the following sample. Note that resource names are decorated with \$partitionname.

```
Dumping Resource Pool:ds-in-template$partition1
Resource Pool:ds-in-template$partition1:dumpPool Current Capacity = 1
Resource Pool:ds-in-template$partition1:dumpPool dumping available resources,
#entries = 1
Resource Pool:ds-in-template$partition1:dumpPool available[0] =
autoCommit=true,enabled=true,isXA=false,isJTS=false,vendorID=100,connUsed=false,doIni
t=false,'null',destroyed=false,poolname=ds-in-
template$partition1,appname=null,moduleName=null,connectTime=862,dirtyIsolationLevel=
false,initialIsolationLevel=2,infected=false,lastSuccessfulConnectionUse=0,secondsToT
rustAnIdlePoolConnection=10,currentUser=null,currentThread=null,lastUser=null,current
Error=null,currentErrorTimestamp=null,JDBC4Runtime=true,supportStatementPoolable=true
,needRestoreClientInfo=false,defaultClientInfo={},supportIsValid=true
Resource Pool:ds-in-template$partition1:dumpPool dumping reserved resources,
#entries = 0
Resource Pool:ds-in-template$partition1:dumpPool # dead resources = 0
Dumping Resource Pool: ds-in-template$partition1 complete
Dumping Resource Pool:ds$partition1
Resource Pool:ds$partition1:dumpPool Current Capacity = 1
Resource Pool:ds$partition1:dumpPool dumping available resources, #entries = 1
Resource Pool:ds$partition1:dumpPool available[0] =
autoCommit=true,enabled=true,isXA=false,isJTS=false,vendorID=100,connUsed=false,doIni
t=false,'null',destroyed=false,poolname=ds$partition1,appname=null,moduleName=null,co
nnectTime=1277,dirtyIsolationLevel=false,initialIsolationLevel=2,infected=false,lastS
uccessfulConnectionUse=0,secondsToTrustAnIdlePoolConnection=10,currentUser=null,curre
ntThread=null,lastUser=null,currentError=null,currentErrorTimestamp=null,JDBC4Runtime
=true,supportStatementPoolable=true,needRestoreClientInfo=false,defaultClientInfo={},
supportIsValid=true
Resource Pool:ds$partition1:dumpPool dumping reserved resources, #entries = 0
Resource Pool:ds$partition1:dumpPool # dead resources = 0
Dumping Resource Pool: ds$partition1 complete
Dumping Resource Pool:ds-using-template$partition1
Resource Pool:ds-using-template$partition1:dumpPool Current Capacity = 1
Resource Pool:ds-using-template$partition1:dumpPool dumping available resources,
#entries = 1
Resource Pool:ds-using-template$partition1:dumpPool available[0] =
autoCommit=true,enabled=true,isXA=false,isJTS=false,vendorID=100,connUsed=false,doIni
t=false,'null',destroyed=false,poolname=ds-using-
template$partition1,appname=null,moduleName=null,connectTime=467,dirtyIsolationLevel=
false,initialIsolationLevel=2,infected=false,lastSuccessfulConnectionUse=0,secondsToT
rustAnIdlePoolConnection=10,currentUser=null,currentThread=null,lastUser=null,current
Error=null,currentErrorTimestamp=null,JDBC4Runtime=true,supportStatementPoolable=true
,needRestoreClientInfo=false,defaultClientInfo={},supportIsValid=true
Resource Pool:ds-using-template$partition1:dumpPool dumping reserved resources,
#entries = 0
Resource Pool:ds-using-template$partition1:dumpPool # dead resources = 0
Dumping Resource Pool: ds-using-template$partition1 complete
```

Configuring Partition-Scoped Deployments

To configure partition-scoped deployments:

1. As with nonpartition scoped deployments, from the home page, select **Deployments**, click **Install**, and then locate the EAR or WAR file that you want to deploy.
2. On the first page of the Install Application Assistant, for deployment scope, you can select either the global scope or one of the existing resource group templates or resource groups.
3. After deploying the EAR or WAR file, you can view the associated modules and scope by clicking the associated link in the console. Initially there is no deployment plan.

To create an application deployment plan, Oracle recommends using the WebLogic Server Administration Console which will create the plan for you:

1. Select the deployed data source.
2. Change the configuration.
3. Save the changes.

The console creates the associated deployment plan and specifies the deployment plan name. The following is a sample application deployment plan.

```
<?xml version='1.0' encoding='UTF-8'?>
<deployment-plan xmlns="http://xmlns.oracle.com/weblogic/deployment-plan"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.oracle.com/weblogic/deployment-plan http://
xmlns.oracle.com/weblogic/deployment-plan/1.01/deployment-plan.xsd">
  <application-name>ds-jdbc.xml</application-name>
  <variable-definition>
    <variable>
      <name>JDBCConnectionPoolParams_InitialCapacity_14417323945070</name>
      <value>1</value>
    </variable>
    <variable>
      <name>JDBCConnectionPoolParams_MinCapacity_14417323945071</name>
      <value>1</value>
    </variable>
    <variable>
      <name>JDBCConnectionPoolParams_StatementCacheSize_14417323945072</name>
      <value>10</value>
    </variable>
    <variable>
      <name>JDBCConnectionPoolParams_MaxCapacity_14417323945073</name>
      <value>20</value>
    </variable>
    <variable>
      <name>JDBCConnectionPoolParams_StatementCacheType_14417323945074</name>
      <value>LRU</value>
    </variable>
  </variable-definition>
  <module-override>
    <module-name>ds-jdbc.xml</module-name>
    <module-type>jdbc</module-type>
    <module-descriptor external="false">
      <root-element>jdbc-data-source</root-element>
      <uri>.</uri>
      <variable-assignment>
        <name>JDBCConnectionPoolParams_InitialCapacity_14417323945070</name>
        <xpath>/jdbc-data-source/jdbc-connection-pool-params/initial-capacity</xpath>
      </variable-assignment>
    </module-descriptor>
  </module-override>
</deployment-plan>
```

```

        <name>JDBCConnectionPoolParams_MinCapacity_14417323945071</name>
        <xpath>/jdbc-data-source/jdbc-connection-pool-params/min-capacity</xpath>
    </variable-assignment>
</variable-assignment>
    <name>JDBCConnectionPoolParams_StatementCacheSize_14417323945072</name>
    <xpath>/jdbc-data-source/jdbc-connection-pool-params/statement-cache-size</
xpath>
</variable-assignment>
</variable-assignment>
    <name>JDBCConnectionPoolParams_MaxCapacity_14417323945073</name>
    <xpath>/jdbc-data-source/jdbc-connection-pool-params/max-capacity</xpath>
</variable-assignment>
</variable-assignment>
    <name>JDBCConnectionPoolParams_StatementCacheType_14417323945074</name>
    <xpath>/jdbc-data-source/jdbc-connection-pool-params/statement-cache-type</
xpath>
</variable-assignment>
</module-descriptor>
</module-override>
<config-root>D:\tmp1221\partition1\ds-jdbc.xml\app\plan</config-root>
</deployment-plan>

```

Using Resource Deployment Plans

If you want to override attributes of a resource group template-derived partition data source other than the user, password, or URL values, then you will need to create a resource deployment plan. See [Configuring Resource Deployment Plans: Main Steps and WLST Example](#).

The following is a sample resource deployment plan. Note that the `descriptor-file-path`, `resource-type`, and the `resource` name elements are used to identify where the descriptor resides.

```

<resource-deployment-plan
  xmlns="http://xmlns.oracle.com/weblogic/resource-deployment-plan"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.oracle.com/weblogic/resource-deployment-plan
  http://xmlns.oracle.com/weblogic/resource-deployment-plan/1.0/resource-deployment-
  plan.xsd">
  <variable-definition>
    <variable>
      <name>JDBCConnectionPoolParams_InitialCapacity_14423700625350</name>
      <value>2</value>
    </variable>
    <variable>
      <name>JDBCConnectionPoolParams_MinCapacity_14423700625511</name>
      <value>5</value>
    </variable>
    <variable>
      <name>JDBCConnectionPoolParams_StatementCacheSize_14423700625512</name>
      <value>10</value>
    </variable>
    <variable>
      <name>JDBCConnectionPoolParams_StatementCacheType_14423700625513</name>
      <value>LRU</value>
    </variable>
  </variable-definition>
  <external-resource-override>
    <resource-name>ds-in-template</resource-name>
    <resource-type>jdbc-system-resource</resource-type>

```

```

<root-element>jdbc-data-source</root-element>
<descriptor-file-path>resource-group-templates/rgt/jdbc</descriptor-file-path>
<variable-assignment>
  <name>JDBCConnectionPoolParams_InitialCapacity_14423700625350</name>
  <xpath>/jdbc-data-source/jdbc-connection-pool-params/initial-capacity</xpath>
</variable-assignment>
<variable-assignment>
  <name>JDBCConnectionPoolParams_MinCapacity_14423700625511</name>
  <xpath>/jdbc-data-source/jdbc-connection-pool-params/min-capacity</xpath>
</variable-assignment>
<variable-assignment>
  <name>JDBCConnectionPoolParams_StatementCacheSize_14423700625512</name>
  <xpath>/jdbc-data-source/jdbc-connection-pool-params/statement-cache-size</
xpath>
</variable-assignment>
<variable-assignment>
  <name>JDBCConnectionPoolParams_StatementCacheType_14423700625513</name>
  <xpath>/jdbc-data-source/jdbc-connection-pool-params/statement-cache-type</
xpath>
</variable-assignment>
</external-resource-override>
</resource-deployment-plan>

```

To associate the resource deployment plan with a partition, provide the path to the resource deployment plan on the partition's **Configuration > General** page.

Configuring JDBC Data Sources: Fusion Middleware Control Example

You can use Fusion Middleware Control to configure JDBC overrides in a manner similar to using the WebLogic Server Administration Console but with a different user interface and navigation paths. However, you must use the WebLogic Server Administration Console to configure data source security because it is not currently available in Fusion Middleware Control.

1. From the **WebLogic Domain** menu, select **JDBC Data Sources** to display a list of existing data sources with their associated type, scope, and if applicable, resource group, resource group template and partition names.
2. To edit an existing data source, click its name.
3. To create a new data source, click **Create** and select the type of data source that you want to create.
4. On the **Data Source Properties** page, use the **Scope** menu to specify the data source scope.
5. From the **WebLogic Domain** menu, select **Environments > Domain Partitions**.
6. Click a partition name to edit the partition attributes.

To create JDBC system resource overrides:

1. Click the partition name.
2. From the left-side **Domain Partition** menu, select **Administration > Resource Overrides**. Use this page to view every resource group that is derived from a resource group template along with the resource type and the name of the resource group template. If there is no existing override, the **Has Overrides** column will be blank.

3. Click the **Edit Overrides** icon to create an override. If the **Has Overrides** column has a check mark in it, then click the **Edit Overrides** icon to update the existing overrides.

Configuring JDBC Data Sources: REST Example

All of the JDBC configuration and runtime information is available using REST. See *Administering Oracle WebLogic Server with RESTful Management Services*. This is an example that gets all of the data sources in the partition, `partition1`:

```
curl --user weblogic:password -H X-Requested-By:MyClient \
  -H Accept:application/json \
  -H Content-Type:application/json -X GET \
  http://host:7001/management/weblogic/latest/serverRuntime/\
  partitionRuntimes/partition1/JDBCPartitionRuntime/JDBCDataSourceRuntimeMBeans
```

The following is abbreviated output:

```
{
...
  "items": [
    {
      ....
      "identity": [
        "partitionRuntimes",
        "partition1",
        "JDBCPartitionRuntime",
        "JDBCDataSourceRuntimeMBeans",
        "ds-in-template"
      ],
      "connectionsTotalCount": 1,
      ...
    },
    {
      ...
      "identity": [
        "partitionRuntimes",
        "partition1",
        "JDBCPartitionRuntime",
        "JDBCDataSourceRuntimeMBeans",
        "ds-using-template"
      ],
      ...
    },
    {
      ...
      "identity": [
        "partitionRuntimes",
        "partition1",
        "JDBCPartitionRuntime",
        "JDBCDataSourceRuntimeMBeans",
        "ds"
      ],
      "connectionsTotalCount": 1,
      ...
    }
  ]
}
```

The following shell script creates a new data source, `ds3`, in a resource group, `partition1-rg`, in a partition, `partition1`, using REST:

```

host=myhost
cmd="curl --user weblogic:password \
  -H X-Requested-By:MyClient \
  -H Accept:application/json \
  -H Content-Type:application/json "
echo "Start a config txn"
${cmd} -d '{}' \
-X POST \
  http://${host}:7001/management/weblogic/latest/edit/changeManager/startEdit

echo "\nCreate the JDBCSystemResource - note - can't save the changes yet"
echo "because we still need to set its JDBCResource's name too."
${cmd} -d "{
  name: 'ds3'
}" -X POST \
  http://${host}:7001/management/weblogic/\
latest/edit/partitions/partition1/\
resourceGroups/partition1-rg/JDBCSystemResources?saveChanges=false
echo "\nSet the JDBCSystemResource's JDBCResource's name and let the changes be"
echo "saved automatically."
${cmd} -d "{
  name: 'ds3'
}" -X POST \
  http://${host}:7001/management/weblogic/latest/\
edit/partitions/partition1/resourceGroups/partition1-rg/\
JDBCSystemResources/ds3/JDBCResource
echo "\nSet the Data Source Params"
${cmd} -d "{
  globalTransactionsProtocol: 'EmulateTwoPhaseCommit',
  JNDINames: [ 'ds3' ]
}" -X POST \
  http://${host}:7001/management/weblogic/latest/\
edit/partitions/partition1/resourceGroups/partition1-rg/\
JDBCSystemResources/ds3/JDBCResource/JDBCDataSourceParams
echo "\nSet the Driver Params"
${cmd} -d "{
  driverName: 'oracle.jdbc.OracleDriver',
  password: 'password',
  url: 'jdbc:oracle:thin:@dbhost:1521/otrade'
}" -X POST \
  http://${host}:7001/management/weblogic/latest/\
edit/partitions/partition1/resourceGroups/partition1-rg/\
JDBCSystemResources/ds3/JDBCResource/JBCDriverParams
echo "\nSet the Properties Params"
${cmd} -d "{
  value: 'dbuser',
  name: 'user'
}" -X POST \
  http://${host}:7001/management/weblogic/latest/\
edit/partitions/partition1/resourceGroups/partition1-rg/\
JDBCSystemResources/ds3/JDBCResource/JBCDriverParams/properties/properties
echo "\nActivate the changes"
${cmd} -d '{}' \
-X POST \
  http://${host}:7001/management/weblogic/latest/edit/changeManager/activate

```

Configuring JDBC Data Sources: Related Tasks and Links

The following references provide additional information on WebLogic Server JDBC data sources, configuring data sources using the Oracle WebLogic Server Administration Console, and an example of configuring JDBC system resources using REST.

- WebLogic Server JDBC Data Sources in *Administering Oracle WebLogic Server with Fusion Middleware Control*.
- Configure Data Sources in *Oracle WebLogic Server Administration Console Online Help*.
- Configuring System Resources and Creating Partition-Scoped System Resources in *Administering Oracle WebLogic Server with RESTful Management Services*.

15

Configuring Messaging

Learn how Oracle WebLogic Server Multitenant (MT) supports messaging, including persistent stores (file and JDBC stores), JMS servers, Store-and-Forward (SAF) agents, path services, messaging bridges, JMS system modules and JMS application modules, and JMS connection pools.

This chapter also describes approaches for accessing partitioned JMS resources from other partitions in the same WebLogic Server instance or cluster, and from remote client or server JVMs.

Note:

WebLogic Server Multitenant domain partitions, resource groups, resource group templates, virtual targets, and Resource Consumption Management are deprecated in WebLogic Server 12.2.1.4.0 and will be removed in the next release.

This chapter includes the following sections:

- [Configuring Messaging: Prerequisites](#)
- [About Messaging Configuration Scopes](#)
- [About Configuration Validation and Targeting Rules](#)
- [Configuring Messaging Components](#)
- [Configuring Partition-Specific JMS Overrides](#)
- [Accessing Partition-Scoped Messaging Resources Using JNDI](#)
- [About Partition Associations in JMS](#)
- [Managing Partition-Scoped Messaging Components](#)
- [Configuring Messaging: Best Practices](#)
- [Configuring Messaging: Limitations](#)
- [Messaging Resource Group Migration](#)

Configuring Messaging: Prerequisites

Prior to configuring JMS in a multitenant environment, you must complete several prerequisite steps.

It is assumed that you are familiar with and have already created:

- A virtual target. See [Configuring Virtual Targets](#).
- A domain partition. See [Configuring Domain Partitions](#).

- A security realm that is specific to the partition, if necessary. See [Configuring Security](#).
- A resource group template, if necessary. See [Configuring Resource Group Templates](#).
- A resource group. See [Configuring Resource Groups](#).

This chapter assumes familiarity with existing WebLogic Server messaging configuration. See:

- *Administering JMS Resources for Oracle WebLogic Server*
- *Administering the WebLogic Persistent Store*
- *Administering the WebLogic Messaging Bridge for Oracle WebLogic Server*
- *Administering the Store-and-Forward Service for Oracle WebLogic Server*
- *Developing Message-Driven Beans for Oracle WebLogic Server*

About Messaging Configuration Scopes

When working with WebLogic Server in nonpartitioned environments, you can configure and deploy JMS artifacts at the domain level. When working in WebLogic Server MT, you can define and deploy JMS artifacts in any one of several scopes.

Examples of JMS artifacts include persistent stores (file or JDBC stores), JMS servers, Store-and-Forward agents, path services, and messaging bridges, which are directly configured in a WebLogic Server domain `config.xml` file using a JMX MBean, such as `PersistentStoreMBean`, `JMSServerMBean`, `SAFAgentMBean`, `PathServiceMBean`, and `MessagingBridgeMBean`.

In addition, JMS resources, such as connection factories and destinations are configured in an external descriptor file called a JMS module. JMS modules are most commonly configured as a JMS system resource (using a `JMSSystemResourceMBean`). Less commonly, JMS modules can be embedded as a standalone or application-scoped XML file that is part of a deployed application (called standalone and application-scoped modules respectively), or indirectly by Java EE 7 connection factory and destination annotations (which have the same basic semantics as external resources defined in an application-scoped module).

When working in WebLogic Server MT, all of the prior JMS artifacts can be defined and deployed in the following scopes:

- Domain-scoped: Using the exact same configuration as in a nonpartitioned WebLogic Server environment
- Resource group-scoped: As part of a resource group that is created at the partition level or at the domain level
- Resource group template-scoped: As part of a resource group template that is created at the domain level

A resource group can optionally inherit a resource group template-scoped JMS configuration. No more than one resource group per partition can reference a particular resource group template, and similarly, no more than one domain level resource group can reference a resource group template.

To summarize, the domain configuration structure for JMS messaging artifacts is as follows:

- Domain-level JMS configuration
- Domain-level resource group with JMS configuration
- Domain-level resource group template with JMS configuration
- Domain-level resource group based on a resource group template
- Partition:
 - Partition-level resource group with JMS configuration
 - Partition-level resource group based on a resource group template

About Configuration Validation and Targeting Rules

Validation and targeting rules ensure that WebLogic Server MT JMS configuration is isolated, self-contained, and easy to manage.

These rules help achieve the following goals:

- A resource group can shut down or migrate independently without causing failures in other resource groups or domain-level resources.
- A resource group template is a fully encapsulated, independent configuration unit without direct dependencies on resource groups, domain configuration, or other resource group templates.
- The same configuration is valid regardless of whether a resource group is single-server targeted, cluster targeted, or not targeted.
- There is no change in behavior for any domain-level configuration that was valid in previous releases. For example, nonresource group or resource group template, domain-level behavior remains unchanged for backwards compatibility.

One basic, high-level rule that helps accomplish these goals is that a JMS configuration MBean may reference only another configuration MBean that is in the same scope. For example, a resource group template-defined JMS server can reference only a store that is also defined in the same resource group template. These rules are enforced by configuration validation checks and by errors and warnings that are logged at runtime.

Configuring Messaging Components

Certain considerations apply to configuring JMS artifacts in a multitenant environment.

- [Configuring JDBC or File Persistent Stores](#)
- [Configuring JMS Servers](#)
- [Configuring Store-and-Forward Agents](#)
- [Configuring Path Services to Support Using Unit-of-Order with Distributed Destinations](#)
- [Configuring Messaging Bridges](#)
- [Configuring JMS System Resources and Application-Scoped JMS Modules](#)

Configuring JDBC or File Persistent Stores

Creating a persistent store is a required step before configuring a JMS server, SAF agent, or path service. This is because resource group and resource group template-scoped JMS servers, SAF agents, and path services must reference an existing persistent store.

Creating a custom file or JDBC persistent store inside a resource group that is either scoped to a domain or to a partition is similar to creating a persistent store at the domain level. However, an additional step is that you must specify the scope. In the Oracle WebLogic Server Administration Console and Oracle Enterprise Manager Fusion Middleware Control, there is a **Scope** menu in the first step of the creation process that lists the available scopes in which to create a persistent store. Using WebLogic Scripting Tool (WLST), you must create the persistent store using the `createPersistentStore` command on the parent MBean (the MBean for the domain, resource group, or resource group template).

The following Distribution Policy and Migration Policy rules apply to all resource group and resource group template-scoped persistent stores:

- A resource group or resource group template-scoped store that will be used to host JMS server distributed destinations or SAF agent imported destinations must specify a `Distributed` Distribution Policy (the default). This setting instantiates a store instance per WebLogic Server instance in a cluster. Furthermore, a resource group or resource group template-scoped store with a `Distributed` Distribution Policy may optionally be configured with an `On-failure` or `Always` Migration Policy.
- A resource group or resource group template-scoped store that will be used by a path service or that will be used to host JMS server standalone (nondistributed) destinations must specify a `Singleton` Distribution Policy. This setting instantiates a single store instance in a cluster. Furthermore, a resource group or resource group template-scoped store with a `Singleton` Distribution Policy must have either `On-failure` or `Always` as its Migration Policy instead of `Off`. `Off` is the default.
- A cluster-targeted store with an `On-failure` or `Always` Migration Policy requires that the cluster be configured with either database leasing or cluster leasing where database leasing is recommended as a best practice.

These policies control the distribution and high availability behavior of stores and any JMS artifacts that target a cluster. See *Simplified JMS Cluster and High Availability Configuration* in *Administering JMS Resources for Oracle WebLogic Server*.

The following are the enforced configuration validation and targeting rules for both file and JDBC stores:

- A resource group or resource group template-level JMS server, SAF agent, or path service must reference a configured store; they cannot reference null.
- A resource group template-scoped JMS server, SAF agent, or path service may reference only a store that is defined in the same resource group template. It cannot reference a store defined at the child resource group level.
- A resource group-scoped JMS server, SAF agent, or path service may reference only a store that is defined in the same resource group, or in the resource group template optionally referenced by the resource group.

- A domain-level JMS server, SAF agent, or path service may reference only a store in the domain scope.

The following are additional rules that are specific to JDBC stores.

- A resource group template-scoped JDBC store may reference only a data source that is in the same resource group template.
- A resource group-scoped JDBC store may reference only a data source that is in the same resource group, or in the resource group template, optionally referenced by the resource group.
- A domain-scoped JDBC store may reference only a data source in the domain scope.

Configuring JMS Servers

Creating a JMS server that is scoped to a domain-level resource group or in a partition is similar to creating a JMS server at the domain level. One additional step is to specify the scope. In the WebLogic Server Administration Console and Fusion Middleware Control, there is a **Scope** menu in the first step of the creation process that lists the available scopes in which to create a JMS server. Using WLST, you must create the JMS server using the `createJMSServer` command on the parent MBean (the MBean for the domain, resource group, or resource group template).

Another required step is to configure the JMS server so that it references a persistent store that is configured in the same scope as the JMS server.

Finally, if the JMS server is going to be used to host distributed destinations, its store must be configured with a `Distributed` Distribution Policy. If the JMS server is going to host standalone (nondistributed) destinations, the store must be configured with a `Singleton` Distribution Policy.

Configuring Store-and-Forward Agents

Creating a Store-and-Forward (SAF) agent that is scoped to a domain-level resource group or in a partition is similar to creating a SAF agent at the domain level. One additional step is to specify the scope. In the WebLogic Server Administration Console and Fusion Middleware Control, there is a **Scope** menu in the first step of the creation process that lists the available scopes in which to create an SAF agent. Using WLST, you must create the SAF agent using the `createSAFAgent` command on the parent MBean (the MBean for the domain, resource group, or resource group template).

Another required step is to configure the SAF agent so that it references a persistent store that is configured in the same scope as the SAF agent. This store must be configured with a `Distributed` Distribution Policy (the default).

Note:

A resource group or resource group template-level SAF agent with a service type `Receiving Only` is not allowed. An exception will be thrown or an error message will be logged on an attempt to set up such a configuration. This mode is specific to outdated JAX-RPC web services reliable messaging. Use JAX-WS RM instead.

Configuring Path Services to Support Using Unit-of-Order with Distributed Destinations

A path service must be configured in a resource group or resource group template if the resource group or resource group template also configures any distributed destinations that will be used to host Unit-of-Order (UOO) messages. In addition, such distributed destinations need to be configured with a Unit-of-Order routing policy set to `PathService` instead of `Hash` because hash-based UOO routing is not supported in a resource group or resource group template scope. Resource group or resource group template-scoped distributed destinations will only use a path service that is configured in the same resource group or resource group template for routing UOO messages. Attempts to send messages to a resource group or resource group template-scoped distributed destination that does not configure a `PathService` Unit-of-Order routing policy will fail with an exception.

Creating a path service that is scoped to a domain-level resource group or in a partition is similar to creating a path service at the domain level. One additional step is to specify the scope. In the WebLogic Server Administration Console and Fusion Middleware Control, there is a **Scope** menu in the first step of the creation process that lists the available scopes in which to create a path service. Using WLST, you must create the path service using the `createPathService` command on the parent MBean (the MBean for the domain, resource group, or resource group template).

Another required step is to configure the path service so that it references a persistent store that is configured in the same scope as the path service. This store must be configured with a `Singleton` Distribution Policy and an `Always` or `On-Failure` Migration Policy.

Configuring Messaging Bridges

Creating a messaging bridge that is scoped to a domain-level resource group or in a partition is similar to creating a messaging bridge at the domain level. One additional step is to specify the scope. In the WebLogic Server Administration Console and Fusion Middleware Control, there is a **Scope** menu in the first step of the creation process that lists the available scopes in which to create a messaging bridge. Using WLST, you must create the messaging bridge using the `createMessagingBridge` command on the parent MBean (the MBean for the domain, resource group, or resource group template).

The following Distribution Policy and Migration Policy rules apply to all resource group or resource group template-scoped messaging bridges:

- Specify a `Distributed` Distribution Policy (the default) on a bridge to cause a cluster-targeted bridge to deploy an instance per server in a cluster. A messaging bridge with a `Distributed` Distribution Policy may optionally also configure an `On-failure` Migration Policy to add support for high availability.
- Specify a `Singleton` Distribution Policy on a bridge to cause a cluster-targeted bridge to limit itself to deploying one instance per cluster. A messaging bridge with a `Singleton` Distribution Policy must have an `On-failure` Migration Policy instead of `Off`. `Off` is the default.

- A cluster-targeted bridge with an `On-failure` Migration Policy requires that the cluster be configured with either database leasing or cluster leasing, where database leasing is recommended as a best practice.

These policies control the high availability behavior and distribution behavior of messaging bridges that target a cluster. For more information about distribution and migration policies, see *Simplified JMS Cluster and High Availability Configuration in Administering JMS Resources for Oracle WebLogic Server*.

The following are the configuration validation rules that are specific to a messaging bridge:

- A resource group template-scoped messaging bridge can reference only messaging bridge destinations in the same scope.
- A resource group-scoped messaging bridge can reference only messaging bridge destinations in the same resource group, or in the resource group template optionally referenced by the resource group.
- A domain-scoped messaging bridge may reference only messaging bridge destinations in the domain scope.

Configuring JMS System Resources and Application-Scoped JMS Modules

Creating a JMS system resource that is scoped to a domain-level resource group or in a partition is similar to creating a JMS system resource at the domain level. One additional step is to specify the scope. In the WebLogic Server Administration Console and Fusion Middleware Control, there is a **Scope** menu in the first step of the creation process that lists the available scopes in which to create a JMS system resource. Using WLST, you must create the JMS system resource using the `createJMSSystemResource` command on the parent MBean (the MBean for the domain, resource group, or resource group template).

Creating an application-scoped JMS module that has a domain-level resource group scope or is in a partition is similar to creating one for the domain level. An application deployment may contain a JMS module file, or an application EAR file that in turn contains JMS module files. One additional step is to specify the resource group or resource group template scope. See [Deploying Applications](#).

Note:

If you create a JMS server and deploy an application that specifies a submodule target to this JMS server all within the same configuration edit session, then the deployment may not succeed. Oracle recommends that you configure the JMS server in a separate edit session.

 **Note:**

Oracle strongly recommends configuring JMS using system resource modules instead of embedding the configuration in application resource modules. Unlike application-scoped configuration, system resource configuration can be dynamically tuned and easily monitored by an administrator or developer using the WebLogic Server Administration Console, WLST, or MBeans.

The following are the configuration validation and targeting rules associated with resources in a resource group or resource group template-scoped JMS module.

Subdeployment Definitions

- A resource group or resource group template-scoped subdeployment only can target nothing (null), a single JMS server, or a single SAF agent.
- A resource group template-scoped subdeployment can reference only a JMS server or SAF agent that is defined in the same resource group template.
- A resource group-scoped subdeployment can reference only a JMS server or SAF agent that is defined in the same resource group or in the resource group template optionally referenced by the resource group.

JMS Module Resources

The following table shows JMS module resource types targeting rules.

Resource Type	Using a Subdeployment	Using Default Targeting
Standalone (Singleton) Destination	May target only a subdeployment which targets a JMS server that in turn references a store with a Singleton Distribution Policy.	Will deploy only if there is a single configured JMS server in the same resource group or resource group template scope that references a Singleton Distribution Policy store. In which case, the destination will deploy on this particular JMS server. JMS servers that reference Distributed Distribution Policy stores are ignored, and JMS servers defined outside the scope, for example, at the domain level or in another resource group or resource group template, are also ignored.

Resource Type	Using a Subdeployment	Using Default Targeting
Uniform Distributed Destination*	May target only a subdeployment which targets a JMS server which in turn references a store with a Distributed Distribution Policy	Will deploy only if there is a single configured JMS server in the same resource group or resource group template scope that references a Distributed Distribution Policy store. In which case, the destination will deploy on this particular JMS server. JMS servers that reference Singleton Distribution Policy stores are ignored, and JMS servers defined outside the scope, for example, at the domain level or in another resource group or resource group template, are also ignored.
SAF Imported Destination	May target only a subdeployment which targets a SAF agent.	Will deploy only when there is a single configured SAF agent in the same resource group or resource group template scope. SAF agents defined outside the scope, for example, at the domain level or in another resource group or resource group template, are also ignored.
Connection Factory	May target any subdeployment.	Will deploy to all WebLogic Server instances that are included in the resource group's target.
Foreign Server	May target only a subdeployment which targets a JMS server which in turn references a store with a Distributed Distribution Policy. Best practice is to use Default Targeting instead	Will deploy to all WebLogic Server instances that are included in the resource group's target.

*** Note:** Resource group or resource group template-scoped uniform distributed topics must specify a `Partitioned Forwarding Policy`. For example, they must be a `Partitioned Distributed Topic (PDT)`. Be aware that the word *Partitioned* in a PDT does not have the same meaning as the word *partition* in a WebLogic Server MT partition. PDTs and WebLogic Server MT partitions are two independent concepts. For information about the trade-offs for using PDTs, see [Configuring Messaging: Limitations](#).

Configuring Partition-Specific JMS Overrides

Resource group template-scoped JMS configuration artifacts might not be complete because they lack or have incorrect values that are specific to partitions that use the resource group template. Each partition may need to have the appropriate override values specified to customize the template-derived values for correct deployment to the partition runtime. Partition-specific, resource group-scoped JMS configuration can be customized on a per-partition basis using resource deployment plans or application deployment plans. In addition, JMS foreign server configuration within a JMS system module can be customized using the `JMSSystemResourceOverrideMBean`.

Resource overriding allows system administrators to customize JMS resources and other resources such as data sources at the partition level. If you create a partition

with a resource group that extends a resource group template, then you can override settings for certain resources defined in that resource group template. If you create a resource group within the partition that does not extend a resource group template and then create resources within this resource group, then you don't need overrides; you can just set partition-specific values for these resources.

Overrides are used mainly when there is a common definition for a resource, such as in a resource group template, that needs each partition that uses the resource to isolate its remotely stored state. For example, the same JMS server, JDBC store, data source, and JMS module configuration can be deployed to multiple partitions in the same cluster by configuring them in a single resource group template and configuring a resource group in each partition to reference the resource group template. The partition resource groups can then be overridden on a per-partition basis to ensure that their respective data sources connect to different databases or to different schemas within the same database.

System administrators can override resource definitions in partitions using the following specific techniques:

- **Resource override configuration MBeans:** A configuration MBean that exposes a subset of attributes of an existing resource configuration MBean. Any attribute set on an instance of an overriding configuration MBean will replace the value of that attribute in the corresponding resource configuration MBean instance. JMS foreign server and related configuration artifacts in a JMS system module can use override MBeans to override the user, password and provider URL settings. If you use override MBeans, you must define a separate override MBean for each corresponding foreign JMS server and related deployment MBeans. Configuration changes to these attributes that are made at runtime after a JMS module has already been deployed require that the partition or JVM be restarted for the changes to take effect.
- **Resource deployment plans:** An XML file that identifies arbitrary configured resources within a partition and overrides attribute settings on those resources. Persistent stores, JMS servers, SAF agents, messaging bridges, bridge destinations, and path services use the `config-resource-override` element in a resource deployment plan, while JMS resources in a JMS system module, such as queues, topics, and connection factories, use the `external-resource-override` element.
- **Partition-specific application deployment plans:** Similar to existing application deployment plans, a plan that allows administrators to specify a partition-specific application deployment plan for each application deployment in a partition. For information about partition-specific application deployment plans, see [Using Partition-Specific Deployment Plans](#).

Administrators can combine any of these resource overriding techniques. The system applies them in the following, ascending order of priority:

- The `config.xml` file and external descriptors, including partition-specific application deployment plans
- Resource deployment plans
- Overriding configuration MBeans

If an attribute is referenced by both a resource deployment plan and an overriding configuration MBean, then the overriding configuration MBean takes precedence.

For more information about overrides, see [Configuring Resource Overrides](#).

Accessing Partition-Scoped Messaging Resources Using JNDI

To access JMS resources in a partition, an application first needs to establish a JNDI initial context to the partition.

After you create a context for a partition, the context object stays in to the partition's namespace so that all subsequent JNDI operations occur within the scope of the partition. When the context is created with a `java.naming.provider.url` property set, JNDI is partition-aware by looking up the partition information from the provider URL value.

The following four different URL types associate the context with a particular partition:

- Specifying no URL.
- Using a URL that specifies a partition's virtual host or URI.
- Using a URL that specifies a partition's dedicated port.
- Using a special `local:` URL. See [Local Cross-Partition Use Cases Using local: URLs or Decorated JNDI Names](#).

In addition, an existing context can be used to reference a resource in another partition by prefixing special scoping strings to JNDI names.

Each of these methods is described in the following sections.

Specifying No URL

An application that is running in a partition on a WebLogic Server instance can access JMS resources in its own local partition simply by creating a local initial context without specifying any provider URL. This approach is the best practice for creating locally scoped contexts.

Specifying a Partition Virtual Host or Partition URI

If a context URL matches a virtual host URL or URI that is configured for a partition, then JNDI creates the context for that partition and all requests from the context are delegated to the partition's JNDI name space.

A JMS application can therefore access a WebLogic Server JMS resource that is running in a different JVM or WebLogic Server cluster using the `t3` or `HTTP` protocol by supplying a URL of the form:

- `t3://virtualhost:port`
- `t3://host:port/URI`

Note:

A misspelled or nonexistent URI may cause a context to scope to the domain level without warning.

Specifying a Dedicated Port URL

It is possible to dedicate a specific port or address to a channel in a partition, in which case the URL format becomes `t3://host:port`.

This is the only supported method for clients prior to release 12.2.1 to interoperate with partition-scoped resources.

See [Configuring Virtual Targets](#).



Note:

This method does not currently support SSL when used for interoperability with previous releases.

Local Cross-Partition Use Cases Using local: URLs or Decorated JNDI Names

An application in one partition can access another partition on the same WebLogic Server instance or in the same cluster using one of the URLs described in the previous section.

However, to support access across partitions that reside on the same server more efficiently without a need to specify a specific host, port, or URI, an application has the following options.

- Create a context with a `local:` protocol URL:
 - `local://` Creates the context on the current partition, which can be either a partition or the domain.
 - `local://?partitionName=DOMAIN` Creates the context on the domain.
 - `local://?partitionName=partition_name` Creates the context on the partition *partition_name*.
- Create a context without specifying a URL, and then prefix an explicit scope when specifying a JNDI name:
 - `domain:<JNDIName>` Looks up the JNDI entry in the domain level.
 - `partition:<partition_name>/<JNDIName>` Looks up the JNDI entry in the specified partition.

About Partition Associations in JMS

Learn about JMS partition associations.

- [Partition Association Between Connection Factories and Their Connections or JMS Contexts](#)
- [Partition Association with Asynchronous Callbacks](#)
- [Connection Factories and Destinations Need Matching Scopes](#)

- [Temporary Destination Scoping](#)

Partition Association Between Connection Factories and Their Connections or JMS Contexts

JMS client connections and JMS contexts are permanently associated with the partition from which their connection factory was obtained, and will not change their partition based on the partition associated with the current thread.

Partition Association with Asynchronous Callbacks

When JMS pushes messages or exceptions to an asynchronous listener, or similarly pushes events to a destination availability listener or an asynchronous send completion listener, the listener's local partition ID (instead of the destination's partition) will be associated with the callback thread. The local partition ID is the partition associated with the thread that created the asynchronous listener.

Connection Factories and Destinations Need Matching Scopes

A connection factory can interact only with a destination defined in the same partition as the connection factory. For example a `QueueBrowser`, `MessageConsumer`/`JMSConsumer`, `TopicSubscriber`, or `MessageProducer`/`JMSProducer` client object can communicate with a destination only if the connection factory that was used to create these client objects was defined in the same partition as the destination. Furthermore, a connection factory can interact with a destination only if it is obtained from the same cluster or server JVM as the connection factory.

Temporary Destination Scoping

Prior to the 12.2.1 release, JMS servers could be deployed only at the domain level and a temporary destination could be hosted only by JMS servers that both:

- Set `Hosting Temporary Destinations` to true (the default).
- Are hosted on the same WebLogic Server instance or in the same cluster as the connection factory used to create the temporary destination.

The behavior for creating a temporary destination in WebLogic Server MT is:

- As in non-MT WebLogic Server, a temporary destination can be hosted only by any JMS server that has `Hosting Temporary Destinations` enabled and that is hosted on the same WebLogic Server instance or in the same cluster as the connection factory used to create the temporary destination.
- If a JMS connection was created using a connection factory that is configured in a resource group or resource group template scope (including domain resource groups), then its temporary destinations will be hosted only by a JMS server that is configured in the same scope.
- If a JMS connection was created using a nonresource group or resource group template-scoped partition-level connection factory, then it is allowed to create temporary destinations on any JMS server from the same partition as the connection factory. The nonresource group or resource group template-scoped partition-level connection factories are simply the default connection factories, for

example the connection factories with JNDI names

`weblogic.jms.ConnectionFactory` OR `weblogic.jms.XAConnectionFactory`.

- If a JMS connection was created using a nonresource group or resource group template-scoped domain-level connection factory, then it is allowed to create temporary destinations on any JMS server at the domain level including JMS servers that are scoped to domain-level resource groups.

If there is no qualified JMS server found within the allowed scope, then an attempt to create a temporary destination returns an exception.

Managing Partition-Scoped Messaging Components

Managing partition-scoped messaging includes runtime monitoring and control, and managing security, transactions, diagnostics, and more.

- [Runtime Monitoring and Control](#)
- [Managing Partition-Scoped Security](#)
- [Managing Transactions](#)
- [Managing Partition and Resource Group Lifecycle Operations](#)
- [Partition-Scoped JMS Diagnostic Image Sources](#)
- [Partition-Scoped JMS Logging](#)
- [Message Lifecycle Logging](#)
- [Admin Helpers](#)
- [File Locations](#)

Runtime Monitoring and Control

All existing messaging runtime MBeans are supported for monitoring and controlling partition-scoped JMS configuration and deployments and are accessible to JMX-based management clients. Partition-scoped JMS runtime MBeans are located under their corresponding `PartitionRuntimeMBean` instances.

For example:

- The `JMSServer`, `Connection`, and `PooledConnection` runtime MBeans are placed in the runtime MBean hierarchy under `serverRuntime/PartitionRuntimes/partition/JMSRuntime`.
- The SAF runtime MBeans are placed in the runtime MBean hierarchy under `serverRuntime/PartitionRuntimes/partition/SAFRuntimeMBean`.
- The messaging bridge and path service runtime MBeans are placed in the runtime MBean hierarchy directly under `serverRuntime/PartitionRuntimes/partition`.

See [Monitoring and Debugging Partitions](#).

Managing Partition-Scoped Security

Security roles and policy definitions related to partition messaging configuration is the responsibility of the WebLogic Server system administrator.

WebLogic Server MT expands upon the traditional WebLogic Server security support in two significant ways:

- Multiple realms: WebLogic Server MT supports multiple active security realms and allows each partition to execute against a different realm.
- Identity domains: An identity domain is a logical namespace for users and groups, typically representing a discrete set of users and groups in the physical data store. Identity domains are used to identify the users associated with particular partitions.

Otherwise, configuring security for partition-scoped messaging is similar to setting up security for domain-level messaging. See [Configuring Security](#).

Managing Transactions

All JTA transactions in a JVM are serviced by a single JTA transaction manager regardless of scope. Partition-scoped XA resource manager names are automatically qualified with their partition name so that the resource managers are uniquely identified to the transaction manager and are managed independently. One example of a resource manager is a persistent store.

For more information about transaction configuration and restrictions, see [Configuring Transactions](#).

Managing Partition and Resource Group Lifecycle Operations

A JMS artifact that is associated with a partition or resource group can be started and shut down by starting and shutting down its partition or resource group. Permissions to perform these operations are automatically supplied to the WebLogic Server system administrator and operator.

Partition-Scoped JMS Diagnostic Image Sources

The messaging component does not support the ability to scope a diagnostic image to a partition. See [Configuring Partition-Scoped Diagnostic Image Capture](#).

Partition-Scoped JMS Logging

Partition-scoped JMS log messages are qualified with the partition ID and name when the domain log format is not configured. For more information about logging, see [Monitoring and Debugging Partitions](#).

Message Lifecycle Logging

Optionally enabled, JMS server and SAF agent message lifecycle logging is placed in locations that are different when these services are scoped to a partition. The logging files are in their partition's directory. Furthermore, the log file names of different runtime JMS server and SAF agent instances of a cluster-targeted JMS server or SAF agent are guaranteed to be replaced with a single interpretation.

The expected new log locations are summarized below when configuring an absolute path, a relative path, or the default.

Scope	Nothing Configured (default)	/<absolute-path>/<file>	/<relative-path>/<file>
Domain Level	<domain-log>/<log-suffix>/<instance>-jms.messages.log	/<absolute-path>/<instance>-<file>	<domain-log>/<relative-path>/<instance>-<file>
Partition	<partition-log>/<log-suffix>/<instance>-jms.messages.log	Same as <relative-path>*	<partition-log>/<relative-path>/<instance>-<file>

* **Note** that partition-scoped configuration treats absolute paths as relative paths.

<domain-log> = <domain>/servers/<wl-server-name>

<partition-log> = <domain>/partitions/<partition-name>/system/servers/<wl-server-name>

<log-suffix> = logs/jmsservers/<configured-name> (for JMS servers)

<log-suffix> = logs/safagents/<configured-name> (for SAF agents)

<instance> =

- <configured-name>, when JMS server or SAF agent is single-server targeted.
- <configured-name>_<wl-server-name>, when cluster-targeted and the data store's Distribution Policy=Distributed.

(Note that an instance keeps its old name even as it migrates from one WebLogic Server instance to another.)

- <configured-name>_01, when cluster-targeted and the data store's Distribution Policy=Singleton.

Admin Helpers

There are two JMS-specific Java administration programming utilities that provide helper methods for configuring and monitoring JMS resources.

The `JMSModuleHelper` utility contains helper methods for locating JMS runtime MBeans (for monitoring) as well as methods to manage (locate/create/delete) JMS module configuration entities (descriptor beans) in a given module.

The `JMSRuntimeHelper` utility provides convenient methods for obtaining the corresponding JMX runtime MBean given a JMS object such as a connection, destination, session, message producer, or message consumer.

In release 12.2.1 and later, the enhanced version of the helpers are provided to handle both domain-scoped and resource group or resource group template-scoped JMS resources.

The existing `JMSRuntimeHelper` utility is enhanced to be partition-aware. When calling a runtime helper method, it is required that the specified JNDI context and the specified JMS object must belong to the same partition (otherwise an exception is thrown).

The enhanced `JMSModuleHelper` utility is scope-aware and contains the following interface and classes:

- `weblogic.jms.extensions.IJMSModuleHelper`—an interface that defines all helper methods.
- `weblogic.jms.extensions.JMSModuleHelper`—the version prior to release 12.2.1 of the JMS module helper, which handles only domain-level JMS resources.
- `weblogic.jms.extensions.JMSModuleHelperFactory`—a factory that creates an instance of a JMS module helper that works in a specific scope given an initial context to the Administration Server, a scope type (domain, resource group or resource group template), and the name of the scope.

The following code demonstrates how to create a JMS module helper for each of the three different scopes:

```
Context ctx = getContext(); // get an initial JNDI context

JMSModuleHelperFactory factory = new JMSModuleHelperFactory();

// create a JMS module helper for domain level

IJMSModuleHelper domainHelper = factory.getHelper(ctx,
IJMSModuleHelper.ScopeType.DOMAIN, null);

// create a JMS module helper for Resource Group "MyResourceGroup"

IJMSModuleHelper rgHelper = factory.getHelper(ctx, IJMSModuleHelper.ScopeType.RG,
"MyResourceGroup");

// create a JMS module helper for Resource Group Template
"MyResourceGroupTemplate"

IJMSModuleHelper rgtHelper = factory.getHelper(ctx,
IJMSModuleHelper.ScopeType.RGT, "MyResourceGroupTemplate");
```

After a JMS module helper instance is created, you can use it to create JMS resources that are scoped to the corresponding scope. For example, the following example code creates a JMS system resource with a JMS queue on JMS server, `MyJMSServer`, in the resource group, `MyResourceGroup`. (It assumes that the JMS server and resource group have already been created.)

```
String jmsServer = "MyJMSServer";

String jmsSystemModule = "MyJMSSystemModule";

String queue = "MyQueue";

String queueJNDI = "jms/myQueue";

rgHelper.createJMSSystemResource(jmsSystemModule, null);

rgHelper.createQueue(jmsSystemModule, jmsServer, queue, queueJNDI, null);
```

File Locations

Persistent stores create a number of files in the file system for different purposes. Among them are file store data files, file store cache files (for file stores with a

DirectWriteWithCache Synchronous Write Policy), and JMS server and SAF agent paging files.

The file location behavior prior to release 12.2.1 remains the same for the domain-scoped persistent stores. This ensures that persistent data is recovered after an upgrade and that it is stored in the expected location. For non-resource group domain-level file locations, see File Locations in *Administering the WebLogic Persistent Store*. For partition-scoped configurations, these files are placed in isolated directories within the partition file system to prevent file collisions among same-named stores in different partitions.

The following summary shows of the location of various files used by the file store system in WebLogic Server MT, where `partitionStem = partitions/<partitionName>/system`.

Store Type	Store Path Not Configured	Relative Store Path	Absolute Store Path	File Name
custom file	<domainRoot>/<partitionStem>/store/<storeName>	<domainRoot>/<partitionStem>/store/<relativePath>/<storeName>	<absolutePath>/<partitionStem>/store/<storeName>	<storeName>NNN.NNN.DAT
cache	\$ {java.io.tmpdir}/WLStoreCache/\${domainName}/<partitionStem>/tmp	<domainRoot>/<partitionStem>/tmp/<relativePath>	<absolutePath>/<partitionStem>/tmp	<storeName>NNN.NNN.CACHE
ejb timers	<domainRoot>/<partitionStem>/store/_WLS_EJBTIMER_<serverName>	<domainRoot>/<partitionStem>/store/_WLS_EJBTIMER_<serverName>	<absolutePath>/<partitionStem>/store/_WLS_EJBTIMER_<serverName>	_WLS_EJBTIMER_<serverName>NNNNN.dat
paging	<domainRoot>/<partitionStem>/paging	<domainRoot>/<partitionStem>/paging/<relativePath>	<absolutePath>/<partitionStem>/paging	<jmsServerName>NNNNNN.TMP <safAgentName>NNNNNN.TMP

The following summary shows how each of the prior store types configure their directory location.

Store Type	Directory Configuration
custom file	The directory configured on a file store.
cache	The cache directory configured on a file store that has a DirectWriteWithCache Synchronous Write Policy.
default ejb timer store	The directory configured on the WebLogic Server default store's configuration. (Partition EJB timer default stores copy their configuration from the default store.)

Store Type	Directory Configuration
paging	The paging directory configured on a SAF agent or JMS server.

Configuring Messaging: Best Practices

Oracle provides recommendations and best practices for beginning JMS users, as well as advanced JMS users, in an MT environment.

- For MT-related known issues, Oracle recommends that all users review Configuration Issues and Workarounds in *Release Notes for Oracle WebLogic Server*.
- If for any reason, newly created or updated JMS resources are not accessible in a running partition, then review the WebLogic Server log files for warning and error log messages. If the server log messages do not provide helpful information, then a partition restart may often resolve the issue. Note that a newly created partition has to be explicitly started before any of the resources is externally accessible.
- The following rules always apply in a resource group and resource group template scope:
 - Use a `Distribution Policy=Singleton` store for path services, and for JMS servers that host standalone destinations.
 - Use a `Distribution Policy=Distributed` store for SAF agents, and for JMS servers that host distributed destinations.
 - Configure cluster leasing in clusters that have:
 - * `Distribution Policy=Singleton` stores or bridges.
 - * `Migration Policy=On-Failure or Always` stores or bridges.

For more general best practices related to using JMS, see Best Practices for JMS Beginners and Advanced Users in *Administering JMS Resources for Oracle WebLogic Server*.

Configuring Messaging: Limitations

WebLogic Server MT does not currently support certain features in JMS or a related component.

- Client SAF forwarding into a partition:
 - The behavior is undefined.
 - Note that there is support for server-side SAF agents to forward into a partition.
- C client access to resource group or resource group template-scoped JMS resources. The behavior is undefined.
- .NET client. An exception is thrown if a .NET client accesses JMS resources in a partition.
- Replicated Distributed Topics (RDT):

- The deployment of a JMS module to a resource group or resource group template that contains Replicated Distributed Topics (RDTs) fails with an exception.
- The default type of uniform distributed topic is configured with a Forwarding Policy of `Replicated`.
- Workarounds include:
 - * Configure a standalone (singleton) topic.
 - * Configure a Partitioned Distributed Topic (PDT).

A PDT is configured by setting its Forwarding Policy to `Partitioned`.

For the advantages and limitations of a PDT, see *Configuring Partitioned Distributed Topics in Administering JMS Resources for Oracle WebLogic Server*.

Note that the word *Partitioned* in a PDT does not have the same meaning as the word *partition* in a WebLogic Server MT partition; PDTs and WebLogic Server MT partitions are two independent concepts.
- * See Replacing a Replicated Distributed Topic in the "Developing Advanced Pub/Sub Applications" chapter of *Developing JMS Applications for Oracle WebLogic Server*.
- Default store
 - Using the WebLogic Server's default store in partitions is not allowed.
 - All JMS servers, SAF agents, and path services in a resource group or resource group template are required to reference a custom store.
- Weighted Distributed Destinations (WDD)
 - The deployment of a JMS module to a resource group or resource group template that contains WDDs fails with an exception.
 - Note that WDDs are deprecated.
- Connection consumer and server session pool
 - An attempt to create a partition-scoped connection consumer or server session pool fails.
 - Note that a best practice is to use a Message Driven Bean (MDB), because MDBs serve a similar purpose to a connection consumer or server session pool.
- Logging Last Resource (LLR) data sources
 - The transaction system does not support the LLR feature in the partition scope.
 - For more information, including a potential workaround, see [Configuring Transactions](#).
- Client interoperability using a dedicated partition channel using SSL
 - Old clients can interoperate only with a partition by configuring a dedicated channel for the partition.
 - This method does not currently support SSL.

Messaging Resource Group Migration

WebLogic Server provides the ability to migrate a resource group to a different server or cluster. All messaging configuration and deployments in a resource group can participate in a resource group's migration.

- Messaging configuration includes file stores, JDBC stores, JMS servers, SAF agents, path services, JMS system resources, and messaging bridges.
- Messaging related deployments include application deployments that contain JMS modules, and EJB and MDB deployments that use JMS resources.

When performing messaging resource group migrations, be aware of the following special considerations:

- WebLogic messaging resource group migration requires that the resource group be shutdown at its source location prior to migration and restarted after the migration completes. If an administrator tries to initiate the migration of a running resource group that contains a messaging configuration, a validation exception will be thrown or displayed with a message directing the administrator to shut down the resource group before migration.
- Non-persistent messaging application and runtime states will not survive a resource group migration.
 - Non-persistent messages will be lost.
 - Clients may get exceptions and need to reconnect during migration. If clients are designed to handle typical JMS failures, they may be able to automatically fail over to the new location. See [Client Failover During Resource Group Migration](#).
- Persistent messaging state data migration may require additional steps.
 - Requires shared storage (database or file) accessible by the source and target locations.
 - For most non-clustered use cases, no additional steps are required.
 - For most cluster use cases, messaging-specific, pre-migration steps are usually required to ensure correct behavior.

See [Resource Group Migration with Persistent Data](#).

- Migration of Message Driven Beans that work with topics may require additional steps. See [Migrating Message Driven Beans \(MDBs\)](#).
- Migration of applications that integrate third party JMS providers may require additional steps. See [Global Transaction Considerations With Third Party JMS](#).

The remainder of this document references three types of messaging services:

- Non-clustered services: services targeted to a non-clustered (standalone) Managed Server
- Cluster singleton services: services targeted to a cluster that have a `Singleton Distribution Policy`.
- Cluster distributed services: services targeted to a cluster that have a `Distributed Distribution Policy`.

A JMS service's distribution policy is configured via a `StoreMBean` or a `MessagingBridgeMBean`, and a messaging service, such as a JMS server, an SAF agent, or a path service, that references a store will inherit its policy from the store MBean configuration. See *Simplified JMS Cluster and High Availability Configuration in Administering JMS Resources for Oracle WebLogic Server*.

The following table summarizes the supported messaging resource group migration scenarios.

Messaging Service	Non-Clustered	Cluster Singleton	Cluster Distributed
JMS Servers and Destinations	Configuration and persistent data	Configuration and persistent data	Configuration only
SAF Agents and Imported Destinations	Configuration only	N/A	Configuration only
Path Services	Configuration and persistent data	Configuration only	Configuration only
Messaging Bridges	Yes	Yes	Yes, with limitations in durable topic cases. See Migrating Persistent Data for a Cluster Distributed Service .

Resource Group Migration with Persistent Data

The following sections describe procedures for non-clustered, cluster singleton, and distributed service handling of persistent data.

Migrating Persistent Data for a Non-Clustered Service

Non-clustered messaging persistent state (messages, durable subscriptions, SAF data, and UOO data) can be safely migrated and processed on the target location with very few exceptions, as described below:

- Additional steps are required to migrate a resource group that contains messaging services from a non-clustered Managed Server to a cluster or vice versa. Prior to such a migration, process all messages, complete all pending transactions, shut down the resource group, and then delete all files and database tables.
- Additional steps are required when migrating a store-and-forward agent. See [Migrating Store-and-Forward Messages](#).
- If a domain level resource group has a file store with an undefined store path, then the path will change after migration because the generated path embeds the current WebLogic Server instance name. For any such store, you must move the files to the new location after the resource group is shutdown on the source location, and before it is restarted on the target location. Oracle recommends that you always specify a store path when configuring a file store.

Migrating Persistent Data for a Cluster Singleton Service

When migrating a cluster singleton service's persistent data, the considerations are the same as the non-clustered use cases, described in [Migrating Persistent Data for a Non-Clustered Service](#), with one additional exception. If a path service is configured,

the store file or table must be deleted after processing all distributed destination messages, completing all transactions, and shutting down the resource group at its original location but before starting the resource group in its new location.

Migrating Persistent Data for a Cluster Distributed Service

Resource group migration requires additional caution in use cases that involve persistent distributed services, for example, distributed destinations and imported destinations. Persistent data associated with such a service can *not* be safely migrated when its hosting resource group migrates. If messages cannot be lost, they must all be processed prior to starting the migration so that no messages remain in any distributed or imported destinations.

In addition, it is important to delete and remove all persistent store files or tables that are associated with such a resource group before migration, even after all the persistent messages are consumed and acknowledged. For example, without a thorough cleanup, durable subscriptions might be abandoned but continue to accumulate messages on the target location, which may cause the server to run out of memory. As another example, UOO messages may be routed to a non-existent location.

The following are detailed considerations in the key cluster distributed messaging use cases:

- Persistent messages in imported destinations and distributed destinations will not be available at the target location. If messages cannot be lost, they must all be processed prior to starting the migration so that no messages remain in any distributed or imported destinations.
- Stores that are cluster-targeted with the Distribution Policy set to `Distributed`, cannot be safely migrated unless you remove all the files or drop the JDBC tables.
- Additional steps are required when migrating a store-and-forward agent. See [Migrating Store-and-Forward Messages](#).
- If you have a remote store-and-forward (SAF) agent that forwards to a distributed destination that you are migrating, additional steps are required on the SAF agent when you migrate the remote distributed destination. See [Migrating Store-and-Forward Messages](#).
- Distributed durable bridges that forward from any topic should be migrated with caution. The corresponding durable subscriptions generated at the original location may be abandoned because the subscription names will be different at the new location, and so the original subscriptions may still accumulate messages on the source topic. Administrators must ensure that the subscriptions that were generated by such a bridge running at the original location are deleted during the migration.
- Distributed destinations that service persistent UOO messages usually use a path service. The path service's store tables or files must be deleted prior to migration to ensure that new UOO messages at the new location are correctly routed.
- With distributed stores, pending global transactions that started before a resource group migration, may not resolve after the migration because the XA resource name of the store instances change after migration. Oracle recommends that administrators make sure that there are no ongoing transactions before performing a resource group migration in a cluster.

Migrating Store-and-Forward Messages

Store-and-forward messages involve two components, an SAF agent that stores and forwards messages and a final WebLogic JMS destination. These are often deployed to separate WebLogic server instances, clusters, or domains, and therefore, you can migrate the two components together or independently.

If you have a remote SAF agent that forwards with exactly-once QOS to a distributed destination, additional steps are required on the SAF agent when you migrate the remote distributed destination. This is because SAF messages may not be able to be forwarded after the migration, which may in turn block subsequent messages from being forwarded. To prevent this situation, Oracle recommends that you empty the SAF queue before migrating the remote distributed destination by doing the following:

1. Pause incoming new messages on the SAF agent. See Controlling Message Operations on Destinations in *Administering JMS Resources for Oracle WebLogic Server*.
2. Wait for all pending messages to be successfully forwarded.

If you are migrating an SAF agent itself that handles exactly-once forwarding, similar steps are recommended regardless of whether it is non-clustered or clustered. This is true even if the SAF agent handles only non-persistent messages. Oracle recommends that you empty the SAF agent's imported destinations (as described above), or delete all the store files or JDBC tables that are associated with the SAF agent before migrating it.

Migrating Message Driven Beans (MDBs)

When an MDB's source destination is a queue, the MDB can be safely migrated. Similarly, an MDB can be safely migrated when an MDB consumes from a topic and its `subscription-durability` is set to `NonDurable`.

Additional caution is required when you migrate an MDB that works with a topic and its `subscription-durability` is set to `Durable`, regardless of whether the MDB or topic is hosted on a cluster or a non-clustered server and regardless of whether the MDB's source destination is hosted in the same cluster or server location as the MDB itself.

In detail:

- You may be able to safely migrate such a durable subscription topic MDB only if it is configured with `TopicMessagesDistributionMode=Compatibility` mode and `generate-unique-client-id=false`, or it is configured with `TopicMessagesDistributionMode=one-copy-per-application` mode. See Deployment Elements and Annotations for MDBs in *Developing Message-Driven Beans for Oracle WebLogic Server*.
- For all other durable topic MDB use cases, `TopicMessageDistributionMode=one-copy-per-server`, or `TopicMessageDistributionMode=Compatibility` and `generate-unique-client-id=true`, the MDB's subscription name contains the local server name, and migrating such MDBs will cause their original durable subscriptions to be abandoned. Not only will the unprocessed messages that are associated with an abandoned durable subscription be lost, also an abandoned durable subscription will continue to accumulate messages, which consumes system memory and eventually may take the system down. As a result, migrating such a durable MDB requires additional caution even if it connects to a non-

clustered, cluster singleton, distributed topic, or third party topic. Administrators must ensure that the subscriptions that were generated by such an MDB running at the original location are deleted during the migration.

When the destination that a MDB listens to is in a separate resource group or partition, you can perform a live migration of the MDB containing resource group but the above considerations for topic MDBs about abandoned durable subscriptions still apply.

Note that when a durable MDB listens to a distributed topic, the associated durable subscriptions may be abandoned when the distributed topic migrates even when the MDB does not migrate. For details about deleting such distributed topic state prior to completing a migration, see [Migrating Persistent Data for a Cluster Distributed Service](#).

Global Transaction Considerations With Third Party JMS

It is important to complete all pending, also known as "in doubt", transactions before migrating a Java EE application that integrates with a third party JMS provider. (In this case, a third party JMS provider is a provider that is neither WL JMS or AQ JMS).

When a Java EE application deployed on a WebLogic Server instance or cluster integrates a third party JMS provider with global transactions, an internal XA resource name is formed under which a foreign XA resource is registered with a WebLogic transaction manager. Those names often contain the local WebLogic Server instance name, which may change after such an application migrates as part of a resource group migration. This may cause in-doubt transactions that cannot be resolved after a migration because the transaction managers will try to resolve them using the original name.

Client Failover During Resource Group Migration

In order to work seamlessly with remote JMS resources that can migrate with their containing resource group, JMS client code needs to follow the known best practice of closing all JMS and JNDI connections and reconnecting to JMS after a JMS failure. See Best Practices for JMS Beginners and Advanced Users in *Administering JMS Resources for Oracle WebLogic Server*. Note that certain containers and services can do this automatically on a client's behalf, namely MDBs, in-bound SOA JMS adapters, and messaging bridges. In addition, such a JMS client needs to ensure that the reconnection logic uses a URL that will resolve to the new location.

When a JMS resource is migrated in a resource group migration, a client may get an exception via exception listener callbacks, or get an exception on a sync call. The client then needs to reestablish connectivity using a new URL that points to the new location of the resource, or using the original URL if an address remapping capability is used during the migration. Examples of DNS remapping include a direct change to a DNS mapping or by using Oracle Traffic Director TCP proxy. When re-establishing, the client will need to retrace the steps it used when establishing its initial connectivity: namely obtaining the initial context, destination lookup, connection factory lookup, connection creation, session creation, producer creation, and finally consumer creation.

Note that JMS client failover behaviors described in the following sections apply to not only standalone clients, but remote or local Java EE applications, such as MDB, and SOA JMS adapter, and messaging bridge as well.

Manual Failover

When no address mapping is used, a client that connects to the original location before migration will need to perform the following steps after the resource group that it accesses migrates:

- Close the existing initial context and JMS objects
- Change the URL to point to the new location
- Reestablish a new initial context using the new URL
- Use the initial context to lookup JMS destinations and connection factories
- Use the connection factories and destinations to connect to JMS

A client that connects only to a WebLogic Sever instance after the migration has to use a URL that points to the new location.

Using Oracle Traffic Director TCP Proxy

The Oracle Traffic Director TCP proxy mechanism provides a mapping capability for a WebLogic Server's hosting host and port to an Oracle Traffic Director proxy's host and port.

In order to use the Oracle Traffic Director capability, you must configure the Oracle Traffic Director mappings for a partition that hosts any to-be-migrated resource groups, the partition's virtual target to use the Oracle Traffic Director proxy's host name, and the applications to use the Oracle Traffic Director proxy's URL to establish an initial context when they connect to any to-be-migrated resource group resources. See [Creating a TCP Proxy](#) in *Oracle Traffic Director Administrator's Guide*.

After you configure the Oracle Traffic Director proxy and the resource group virtual target to use Oracle Traffic Director, an application client is able to reconnect and fail over to the new location after a resource group migration with less manual intervention if the client does one of the following:

- Reestablishes a new initial context
- Creates a new JMS connection after the migration using a connection factory stub that it obtained before the migration

With the existing Oracle Traffic Director support, the only manual step that is required is to re-map the target URL to the proxy's URL and restart the proxy instances after migration. The restart is required to ensure that all the existing connections to the old server are dropped. Without the restart, new clients that connect after migration will be routed to the new location, but the clients that were connected before the migration may not fail over to the new location.

16

Configuring and Programming JNDI

Learn how to configure foreign Java Naming and Directory (JNDI) providers in Oracle WebLogic Server Multitenant (MT), including programming JNDI in a partitioned domain, resource-scoped, object-based partition association, binding and obtaining partition information, accessing resources over partitions, and clustered JNDI in partitions.

Note:

WebLogic Server Multitenant domain partitions, resource groups, resource group templates, virtual targets, and Resource Consumption Management are deprecated in WebLogic Server 12.2.1.4.0 and will be removed in the next release.

This chapter includes the following sections:

- [Configuring Foreign JNDI Providers: Overview](#)
- [Configuring Foreign JNDI Providers: Prerequisites](#)
- [Configuring Foreign JNDI Providers: Main Steps](#)
- [Creating Foreign JNDI Providers: WLST Example](#)
- [Programming JNDI in a Partitioned Environment](#)
- [Creating Foreign JNDI Providers: Related Tasks and Links](#)

Configuring Foreign JNDI Providers: Overview

WebLogic Server provides a foreign JNDI API that lets you access objects on a remote JNDI tree without having to connect directly to the remote tree. In a partitioned environment, you can access objects on a JNDI tree either locally (in another partition on the same machine) or remotely.

By creating and configuring a foreign JNDI provider with the properties of the other partitions, you can look up and use an object that exists outside of a given partition. The properties that you set for the foreign JNDI provider are used to create a new context that internally does the actual lookup and bind operations.

Configuring Foreign JNDI Providers: Prerequisites

Prior to creating and configuring a foreign JNDI provider for a partition, you must perform certain prerequisite steps.

1. Create one or more virtual targets. See [Configuring Virtual Targets](#).
2. Create a resource group in the partition to use as the scope for the foreign JNDI provider in the partition. When creating the resource group, select one or more

available virtual targets for the resource group. See [Creating Resource Groups: Main Steps and Examples](#).

Configuring Foreign JNDI Providers: Main Steps

The procedure for configuring a foreign JNDI provider for a partition is the same as configuring one for a domain, with some exceptions.

- Set the scope of the foreign JNDI provider to a resource group in the partition.
- If the foreign JNDI provider exists in a partition on another machine, then set the provider URL as if it were a common remote client. For example, for the t3 protocol, `t3://hostname:port/partition_name`.

To complete the foreign JNDI provider configuration, create one or more foreign JNDI links to associate local JNDI names with JNDI names on remote nodes.

Creating Foreign JNDI Providers: WLST Example

You can use WebLogic Scripting Tool (WLST) to create a foreign JNDI provider and configure the user name, password, provider URL, and the `InitialContextFactory` for the provider. Also, to create a foreign JNDI link for the provider.

```
# Connect to the Administration Server
connect('adminusername','adminpassword','t3://hostname:port')

# Start an edit session
edit()
startEdit()

# Change to the appropriate resource group directory for the partition for which
# you are creating the foreign JNDI provider.
cd('/Partitions/partition_name/ResourceGroups/resource_group_name')
cmo.createForeignJNDIProvider('provider_name')

cd('ForeignJNDIProviders/provider_name')
set('Password','password')
cmo.setUser('username')
cmo.setProviderURL('t3://hostname:port')
cmo.setInitialContextFactory('initialContextFactory')
# Create a foreign JNDI link and configure it
cmo.createForeignJNDILink('link_name')
cd('ForeignJNDILinks/link_name')
cmo.setLocalJNDIName('local_JNDI_name')
cmo.setRemoteJNDIName('remote_JNDI_name')
save()
activate()
```

Programming JNDI in a Partitioned Environment

Learn key points to be aware of when programming JNDI in a partitioned environment versus a nonpartitioned environment.

- [Introduction to Partition-Scoped and Domain-Scoped JNDI Resources](#)
- [Object-Based Partition Association](#)

- [Obtaining the Partition Information of a Context](#)
- [Accessing JNDI Resources Remotely and Across Partitions](#)
- [Clustered JNDI in a Partitioned Environment](#)
- [Life Cycle of a Partitioned JNDI Resource](#)

For additional information about programming JNDI, see *Developing JNDI Applications for Oracle WebLogic Server*.

Introduction to Partition-Scoped and Domain-Scoped JNDI Resources

In WebLogic Server, there is only one global JNDI tree that serves all requests for global JNDI resources. In WebLogic Server MT, however, there is a global JNDI tree for the domain and a global JNDI tree for each partition. JNDI resources in a partition are, by default, available only to the partition itself, although you can access such resources across partitions as described in [Accessing JNDI Resources Remotely and Across Partitions](#). Therefore, by default, applications deployed at the partition level have access only to the JNDI resources within the partition and cannot access JNDI resources for other partitions or the domain.

For example, if you deploy an EJB with the global JNDI name `java:global/foo` to the domain and to multiple partitions on the same server, then this results in `java:global/foo` being bound to the global JNDI tree for each of these partitions and to the global JNDI tree for the domain. This results in the following behavior:

- When you perform a lookup of `java:global/foo` from one partition, by default, JNDI returns the instance that is bound in the global JNDI tree for that partition.
- When you perform a lookup of `java:global/foo` from the domain, JNDI returns the instance that is bound to the global JNDI tree for the domain.

Here are some additional examples:

- `mail.MedRecMail Session` is bound to partitionA, partitionB, and the domain. JNDI lookup requests from an application in partitionA get the session for partitionA, while JNDI lookup requests from an application in partitionB get the session for partitionB. JNDI lookup requests from an application deployed in the domain will get the session for the domain.
- `weblogic.transaction.resources.dsXA` is bound only to partitionB. Lookup requests from applications deployed in partitionB get the `RmiDataSource`. Requests from applications deployed in other partitions or the domain will get a `NameNotFoundException` exception.
- `java:global/wm/default` is bound only at the domain level. Only applications deployed in the domain can access it by default.

Note:

Requests such as lookup or bind to application-scoped JNDI resources are isolated naturally because they are delegated to the application.

Object-Based Partition Association

When you create a JNDI context within a partition, the context object stays in the partition namespace so that all subsequent JNDI operations are done within the context of the partition. When the JNDI context is created, the association to a specific partition is established by the specified provider URL property. If you create the JNDI context with the `java.naming.provider.url` property set, then JNDI is partition-aware by looking up the partition information from the provider URL value. If you set the provider to be the virtual target URL that is configured for the partition, then JNDI creates the context for that partition and all requests from the context are delegated to the partition's JNDI resources.

After the object-based context has been created, its operations are performed using the partition JNDI tree.

Obtaining the Partition Information of a Context

Partition information is bound to the partition global JNDI tree when the partition tree is initialized. You can obtain the partition information of a context by looking up:

- `weblogic.partitionName`, which returns the context-based partition's `partitionName`. This will either be a partition name or `Domain` if it is a domain-scoped context.
- `weblogic.partitionId`, which returns the partition's `partitionId`. This will be zero, 0, if it is a domain or a value greater than 0 if it is a partition.

Accessing JNDI Resources Remotely and Across Partitions

Partition JNDI resources can be accessed from remote, standalone Java code using the WebLogic Server client or code that resides on a remote WebLogic Server instance. This is done by setting the provider URL in the same way as you would do if you were accessing a remote single server JNDI tree.

WebLogic Server JNDI also enhances foreign JNDI providers to allow one partition to declare only a local JNDI name but actually point to other partition JNDI resources. You can configure a link entry to associate a local JNDI name with a partition JNDI resource.

Cross-Partition Authentication

JNDI context authentication is done when the context is being created. WebLogic Server JNDI ensures that the authentication is processed in the partition to which the provider URL is associated, not the current partition. If no provider URL is set when creating the context, then the authentication is processed in the security realm associated with the current partition. The authenticated context is then pushed into the thread context for a permission check on the context in subsequent operations.

Clustered JNDI in a Partitioned Environment

The JNDI tree representing a cluster appears to the client as a single global tree. The tree containing the clusterwide services is actually replicated across each WebLogic Server instance or partition in the cluster.

In a multitenant environment, when setting replicate binding to a JNDI resource, the bind, unbind, and rebind events are exposed to all cluster nodes.

Life Cycle of a Partitioned JNDI Resource

WebLogic Server maintains the life cycle of partition JNDI resources according to the partition life cycle:

- When a partition is created and started, the partition JNDI tree is created with the partition root node and becomes available.
- When a partition is shutting down, the entire partition JNDI tree is deleted.

Creating Foreign JNDI Providers: Related Tasks and Links

- [Configuring Virtual Targets](#)
- [Configuring Resource Groups](#)
- *Developing JNDI Applications for Oracle WebLogic Server*

Configuring Partition Work Managers

Partition Work Managers set thread usage policy among partitions. You can configure them to limit the number of Work Manager threads in each partition, as well as to manage thread usage allocation based on thread usage time for each partition. The importance of regulating the relative thread usage is to provide proper quality of service (QoS) and fairness among various partitions that share the same WebLogic Server instance. Without it, an application from one partition could starve thread resources from other partitions preventing them from functioning properly.

 **Note:**

WebLogic Server Multitenant domain partitions, resource groups, resource group templates, virtual targets, and Resource Consumption Management are deprecated in WebLogic Server 12.2.1.4.0 and will be removed in the next release.

This chapter includes the following sections:

- [Partition Work Managers: Overview](#)
- [Configuring Partition Work Managers: Main Steps](#)
- [Defining Partition Work Managers: WLST Example](#)
- [Configuring Partition Work Managers: Related Tasks and Links](#)

Partition Work Managers: Overview

Partition Work Managers provide resource management within partitions.

Administrators know about the runtime environment and how it will be shared. They configure Partition Work Managers at the domain level and assign them to partitions as they are created. These predefined Partition Work Managers let administrators standardize Work Manager configuration; for example, all partitions with business-critical applications can reference the business-critical Partition Work Manager.

Administrators might also want to customize the Partition Work Manager for a specific partition, or maybe for every partition. In this scenario, they configure the Partition Work Manager within (embedded in) the partition configuration. There is no need to predefine Partition Work Manager configurations for this use case.

You can define a Partition Work Manager in the domain to use with multiple domain partitions, or you can define Partition Work Manager attributes in the domain partition itself for use in that partition only. If no Partition Work Managers are defined, then default values for Partition Work Manager settings are applied.

Partition Work Managers can be used in more than one domain partition. However, a domain partition can be associated with only one Partition Work Manager.

A partition configuration can include one of the following:

- `<partition-work-manager-ref>` to refer to a Partition Work Manager that is configured at the domain level
- `<partition-work-manager>` to embed the Partition Work Manager settings within the partition configuration
- Neither `<partition-work-mananger>` nor `<partition-work-manager-ref>` to apply the default values for Partition Work Manager settings

Configuring Partition Work Managers: Main Steps

Partition Work Managers define a set of policies that limits the usage of threads by Work Managers in partitions only. They do not apply to the domain. Follow these main steps for configuring a Partition Work Manager.

1. Enter a name for the Partition Work Manager.
2. Optionally, enter a fair share value, the desired percentage of thread usage for the partition compared to the thread usage of all partitions. Oracle recommends that the sum of this value for all partitions running in a WebLogic Server domain add up to 100. By default, a partition has a fair share value of 50.
3. Optionally, enter a minimum threads constraint value, the upper limit on the number of standby threads that can be created for satisfying the minimum threads constraints configured in the partition.

A minimum threads constraint guarantees the number of threads that the server allocates to a Work Manager to avoid deadlocks. This could result in a Work Manager receiving more thread use time than its configured fair share, and thus, a partition getting more thread usage time than it should compared to other partitions in the same WebLogic Server instance.

You can optionally provide a limit on the minimum threads constraint value for each partition configured in the WebLogic Server domain. If configured, then this imposes an upper limit on the minimum threads constraint values configured in a partition. If the sum of the configured values of all minimum threads constraints in a partition exceeds this configured value, then a warning message will be logged and WebLogic Server reduces the number of threads that the thread pool allocates for the constraints.

There is no minimum threads constraint limit set on a partition by default.

4. Optionally, enter a maximum threads constraint value, the maximum number of concurrent requests that the self-tuning thread pool can process for a partition at any given time.

A maximum threads constraint value can be useful to prevent a partition from using more than its fair share of thread resources, especially in unusual situations such as when threads are blocked on I/O, waiting for responses from a remote server that is not responding. Setting a maximum threads constraint in such a scenario would help ensure that some threads would be available for processing requests from other partitions in the WebLogic Server instance.

5. Optionally, enter a shared capacity constraint value, the total number of requests that can be present in the server for a partition as a percentage.

The partition-shared capacity for Work Managers limit the number of work requests from a partition. This limit includes work requests that are either running

or queued waiting for an available thread. When the limit is exceeded, WebLogic Server will start rejecting certain requests submitted from the partition. The value is expressed as a percentage of the capacity of the entire WebLogic Server as configured in the `sharedCapacityForWorkManagers` option of the `OverloadProtectionMBean` that constricts the number of requests in the entire WebLogic Server instance. The partition-shared capacity for Work Managers must be a value between 1 and 100 percent.

Defining Partition Work Managers: WLST Example

These sample scripts show how to define Partition Work Managers using WLST.

You can define Partition Work Managers by either configuring a domain-level Partition Work Manager and then associating it with a partition, or defining Partition Work Manager attributes in a domain partition, for use in that partition only.

Configuring Domain-Level Partition Work Managers: WLST Example

The following example creates and configures the domain-level Partition Work Manager, `myPartitionWorkManager`.

```
# Creates a Partition Work Manager at the domain level
edit()
startEdit()
cd('/')
cmo.createPartitionWorkManager('myPartitionWorkManager')
activate()

# Configures the Partition Work Manager
startEdit()
cd('/PartitionWorkManagers/myPartitionWorkManager')
cmo.setSharedCapacityPercent(50)
cmo.setFairShare(50)
cmo.setMinThreadsConstraintCap(0)
cmo.setMaxThreadsConstraint(-1)
activate()
```

Associating Partition Work Managers with Partitions: WLST Example

Partition Work Managers can be used in more than one domain partition. However, a domain partition can be associated with only one Partition Work Manager. The following example associates the domain-level Partition Work Manager, `myPartitionWorkManager` with the partition, `Partition-0`:

```
# Associate a domain-level Partition Work Manager with a partition.
edit()
startEdit()
cd('/Partitions/Partition-0')
cmo.destroyPartitionWorkManager(None)
cmo.setPartitionWorkManagerRef(getMBean('/PartitionWorkManagers/
myPartitionWorkManager'))
activate()
```

Defining Partition Work Manager Attributes in a Partition: WLST Example

The following example defines Partition Work Manager attributes in the domain partition, Partition-1, for use in that partition only:

```
# Defines Partition Work Manager attributes within the partition
edit()
startEdit()
cd('/Partitions/Partition-1')
cmo.createPartitionWorkManager('Partition-1-PartitionWorkManager')

cd('/Partitions/Partition-1/PartitionWorkManager/Partition-1-PartitionWorkManager')
cmo.setSharedCapacityPercent(50)
cmo.setFairShare(50)
cmo.setMinThreadsConstraintCap(0)
cmo.setMaxThreadsConstraint(-1)
activate()
```

In the `config.xml` file, notice the Partition Work Manager element defined in Partition-0 and Partition-1:

```
<partition>
  <name>Partition-0</name>
  <resource-group>
    <name>default</name>
  </resource-group>
  <default-target>VirtualTarget-0</default-target>
  <available-target>VirtualTarget-0</available-target>
  <realm>myrealm</realm>
  <partition-id>318e0d69-a71a-4fa6-bd7e-3d64b85ec2ed</partition-id>
  <system-file-system>
    <root>C:\Oracle\Middleware\Oracle_Home\user_projects\domains\base_domain\
partitions/Partition-0/system</root>
    <create-on-demand>true</create-on-demand>
    <preserved>true</preserved>
  </system-file-system>
  <partition-work-manager-ref>myPartitionWorkManager</partition-work-manager-ref>
</partition>
<partition>
  <name>Partition-1</name>
  <resource-group>
    <name>default</name>
  </resource-group>
  <default-target>VirtualTarget-1</default-target>
  <available-target>VirtualTarget-1</available-target>
  <realm>myrealm</realm>
  <partition-id>8b7f6bf7-5440-4edf-819f-3674c630e3f1</partition-id>
  <system-file-system>
    <root>C:\Oracle\Middleware\Oracle_Home\user_projects\domains\base_domain\
partitions/Partition-1/system</root>
    <create-on-demand>true</create-on-demand>
    <preserved>true</preserved>
  </system-file-system>
  <partition-work-manager>
    <name>Partition-1-PartitionWorkManager</name>
    <shared-capacity-percent>50</shared-capacity-percent>
```

```
<fair-share>50</fair-share>
<min-threads-constraint-cap>0</min-threads-constraint-cap>
<max-threads-constraint>-1</max-threads-constraint>
</partition-work-manager>
</partition>
<partition-work-manager>
  <name>myPartitionWorkManager</name>
  <shared-capacity-percent>50</shared-capacity-percent>
  <fair-share>50</fair-share>
  <min-threads-constraint-cap>0</min-threads-constraint-cap>
  <max-threads-constraint>-1</max-threads-constraint>
</partition-work-manager>
</domain>
```

Configuring Partition Work Managers: Related Tasks and Links

The referenced information provides steps for configuring Partition Work Managers using Oracle Enterprise Manager Fusion Middleware Control (FMWC) and the Oracle WebLogic Server Administration Console.

- [WebLogic Server Partition Work Managers in *Administering Oracle WebLogic Server with Fusion Middleware Control*](#)
- [Partition Work Managers in *Oracle WebLogic Server Administration Console Online Help*](#)

18

Migrating a WebLogic Server Domain to a Domain Partition

Learn how to use the Domain to Partition Conversion Tool (D-PCT) to migrate existing applications and resources from a non-multitenant WebLogic Server domain to a multitenant domain partition.

Note:

WebLogic Server Multitenant domain partitions, resource groups, resource group templates, virtual targets, and Resource Consumption Management are deprecated in WebLogic Server 12.2.1.4.0 and will be removed in the next release.

This chapter includes the following sections:

- [Migrating a WebLogic Server Domain to a Domain Partition: Overview](#)
- [Migrating a WebLogic Server Domain to a Domain Partition: Prerequisites](#)
- [Migrating a WebLogic Server Domain to a Domain Partition: Main Steps](#)
- [Migrating a WebLogic Server Domain to a Domain Partition: Limitations and Considerations](#)

Migrating a WebLogic Server Domain to a Domain Partition: Overview

The Domain to Partition Conversion Tool (D-PCT) lets you migrate an existing WebLogic Server 10.3.6, 12.1.2, or 12.1.3 domain to a WebLogic Server 12.2.1.x domain partition. D-PCT consists of two tools: an export tool, which is used on the WebLogic Server installation that hosts the source domain; and an import tool, which is used on the target WebLogic Server 12.2.1.x installation.

You can use D-PCT to create and configure partitions, resource groups, and resource group templates. By default, D-PCT moves all the applications, libraries, and resources to the new partition. Optionally, it also provides a mechanism for selecting individual applications, libraries, and resources.

Note:

The domain's cluster and server configurations are not applicable to a partition. Therefore, they are not mapped to the new 12.2.1.x partition.

New features of the Domain to Partition Conversion Tool

The Domain to Partition Conversion tool has introduced new features in 12.2.1.1.0. The following are the significant changes:

- D-PCT now provides support for targeting resources and applications to multiple virtual targets.

In the previous release, you could use D-PCT to target all the resources in the source domain to only one virtual target. As of 12.2.1.1.0, you can use this tool for targeting resources and applications to multiple WebLogic Servers and clusters using multiple virtual targets, where each virtual target is associated to a single WebLogic Server instance or a cluster. All application deployments and resources targeted to that particular WebLogic Server instance or cluster in the source domain corresponds to a resource group in the target domain.

In order to accomplish this, the structure of the JSON file generated by D-PCT in 12.2.1.1.0 is modified. Therefore, any archive generated from the initial release of D-PCT cannot be imported into a 12.2.1.1.0 domain. An error will be reported during the import operation. For existing domain archives, you must regenerate a new domain archive using the D-PCT 12.2.1.1.0 tooling to enable it to be imported into the 12.2.1.1.0 domain.

- D-PCT has enhanced support for JMS resources targeted to clusters and migratable targets. See [Considerations and Limitations for Importing a Domain with JMS Resources](#).
- The JSON file generated by D-PCT in 12.2.1.1.0 gives more flexibility to the users, by exposing the JDBC System Resources, SAF Agents, Mail Sessions and JDBC Stores. These resources are available as JSON objects, `jdbc-system-resource`, `saf-agent`, `mail-session`, and `jdbc-store` respectively.

Migrating a WebLogic Server Domain to a Domain Partition: Prerequisites

Before you configure D-PCT, you must fulfill certain prerequisite requirements.

- The target domain must be configured on a WebLogic Server 12.2.1.x installation.
- You must download and install JDK 8 on the host machine of the source domain. This is required for running the export tool on WebLogic Server 10.3.6.
- The target WebLogic Server 12.2.1.x domain must be up and running.
- For a straightforward conversion from a domain to a partition and to ensure correct operation of the import tool to create a virtual target based on the JavaScript Object Notation (JSON) file, it is important that the target and source domains have identical cluster and server names.

By default, the import tool creates virtual targets with the specified name and target in the JSON file. However, Oracle recommends that you manually configure virtual targets before the import operation. If you preconfigure virtual targets before using the import tool, then you must ensure that you modify the `partition` attribute of the JSON file, where you must specify the name of the existing virtual target that you created at the target domain for this partition to use.

- Before using the import tool to create a new partition, you must ensure that the new domain is configured the same as the source domain with regard to servers, clusters, virtual targets, and security realms. For information about importing partitions, see [Exporting and Importing Partitions](#).

Migrating a WebLogic Server Domain to a Domain Partition: Main Steps

The main steps for migrating a WebLogic Server domain to a domain partition include preparing and exporting the WebLogic Server domain applications; if needed, using the JSON file to override application default values; then importing the applications to the newly created domain partition.

- [Preparing to Export the WebLogic Server Domain Applications Environment](#)
- [Exporting the WebLogic Server Domain Applications Environment](#)
- [Using the JSON File to Override Defaults During Import](#)
- [Importing an Applications Archive File to a Domain Partition](#)
- [Monitoring the Export and Import Operation using Reports](#)

Preparing to Export the WebLogic Server Domain Applications Environment

Perform the following tasks to prepare to export the WebLogic Server domain application environment:

1. Navigate to the directory where the D-PCT tool zip file is located. The domain export tool is bundled as part of the WebLogic Server 12.2.1.x distribution and included as part of the product install. The D-PCT tool zip file is available on your host machine at `MW_HOME/wlserver/common/dpct/D-PCT-12.2.1.1.0.zip`.
2. Unzip the `D-PCT-12.2.1.1.0.zip` file. You can unzip this file to any preferred directory. Oracle recommends creating a separate directory for the export tool. You must unzip the export tool zip distribution into this directory, and then run the export tool script from it as well.

This zip distribution consists of the following files:

- `README.txt` - a file that contains documentation for installing and running the Domain to Partition Conversion Tool (D-PCT).
- `exportDomainForPartition.sh` - a UNIX script that executes at the source WebLogic Server domain to export the domain configuration into an archive, which can be imported to a domain partition.
- `export-domain.sh` - an alternate UNIX script that executes at the source WebLogic Server domain to export the domain configuration into an archive, which can be imported to a domain partition.
- `exportDomainForPartition.cmd` - a Windows script that executes at the source WebLogic Server domain to export the domain configuration into an archive, which can be imported to a domain partition.

- `export-domain.cmd` - an alternate Windows script that executes at the source WebLogic Server domain to export the domain configuration into an archive, which can be imported to a domain partition.
- `com.oracle.weblogic.management.tools.migration.jar` - a JAR file that contains WLST scripts and Java classes used for exporting the source domain to a domain archive file.

 **Note:**

The classes in `com.oracle.weblogic.management.tools.migration.jar` are built with JDK 8. Therefore, to run the export tool on WebLogic Server 10.3.6, the export script must be run with JDK 8 which might not be the JDK version used normally with WebLogic Server 10.3.6 installation. Before running the export script, ensure that you set the `JAVA_HOME` variable accordingly.

3. Create a key file that contains a string to use as the encryption key to encrypt attributes in the partition archive file. The path must be reachable by the Administration Server. The size of the key string must range between 1 to 32 characters.

Exporting the WebLogic Server Domain Applications Environment

To export the WebLogic Server domain applications, you must run the export script on the host machine where the source WebLogic Server domain Administration Server resides.

- The syntax for the arguments passed to this script on UNIX is:

```
exportDomainForPartition.sh ORACLE_HOME DOMAIN_HOME [keyFile] [TOOL_JAR]
[app_name] [INCLUDE_APP_BITS={true|false}] [WL_HOME]
```

or

```
export-domain.sh -oh {ORACLE_HOME} -domainDir {DOMAIN_HOME} [-keyFile {KEYFILE}]
[-toolJarFile {TOOL_JAR}] [-appNames {APP_NAMES}] [-includeAppBits
{INCLUDE_APP_BITS}] [-wlh {WL_HOME}]
```

- The syntax for the arguments passed to this script on Windows is:

```
exportDomainForPartition.cmd ORACLE_HOME DOMAIN_HOME [keyFile] [TOOL_JAR]
[app_name] [INCLUDE_APP_BITS={true|false}] [WL_HOME]
```

or

```
export-domain.cmd -oh {ORACLE_HOME} -domainDir {WL_DOMAIN_HOME} [-keyFile
{KEYFILE}] [-toolJarFile {TOOL_JAR}] [-appNames {APP_NAMES}] [-includeAppBits
{INCLUDE_APP_BITS}] [-wlhL_HOME}]
```

Before running the script on Windows, perform the following tasks:

1. Open a command shell and change to the directory where you unzipped the export tool distribution.
2. Set the `JAVA_HOME` environment variable to the path of your JDK 8 installation as shown in the following example:

```
C:\> set JAVA_HOME=C:\jdk1.8.0
```

 **Note:**

Oracle recommends that you do not make any configuration changes at the source domain during the export operation.

This script accepts the arguments described in the following table.

Argument	Description
<i>ORACLE_HOME</i>	The name of the Oracle home directory where WebLogic Server is installed. This argument can be specified as the name=value pair. For example, <code>ORACLE_HOME=/Oracle/Middleware/Oracle_Home</code> .
<i>DOMAIN_HOME</i>	The full path to the source WebLogic Server domain. This argument can be specified as the name=value pair. For example, <code>DOMAIN_HOME=/Oracle/Middleware /Oracle_Home/user_projects/domains/medrec</code> .
<i>keyFile</i>	Optional. The full path to a file containing a string to use as the encryption key to encrypt attributes in the partition archive file. The Administration Server must have access to this path. The default value is None.
<i>TOOL_JAR</i>	Path to the <code>com.oracle.weblogic.management.tools.migration.jar</code> file. This argument is optional if the JAR file is located in the same directory as the <code>exportDomainForPartition.sh</code> or the <code>export-domain.sh</code> script.
<i>app_name</i>	Optional. The list of application names to export. On Windows, the list of applications must be specified as <code>[myapp1+myapp2]</code> ; on Unix, it must be specified as <code>['myapp1', 'myapp2']</code> . If not specified, all applications in the domain are exported.
<i>INCLUDE_APP_BITS</i>	Optional. A flag to indicate whether the application binary files are included in the archive file. This value defaults to <code>true</code> , which means the binary files are included. If <code>false</code> , then those files are excluded.

Argument	Description
<code>WL_HOME</code>	Optional. The full path to the directory where WebLogic Server is installed. This value must be specified <i>only</i> if WebLogic Server 10.3.6 is installed in a directory other than the default location, <code>ORACLE_HOME/wlserver_10.3</code> . This value is used to determine the location of the <code>wlst.sh</code> and <code>wlst.cmd</code> scripts.

 **Note:**

While specifying optional arguments for the `exportDomainForPartition.cmd` and `exportDomainForPartition.sh` scripts, ensure that you specify `None` for any optional argument that you do not wish to supply, before passing the next optional argument. If you pass an optional argument with a valid value without specifying `None` for the previous optional argument, then the value you specified may get wrongly assigned to the previous argument. However, you do not need to specify `None` for the optional argument that appears after the last optional argument with a valid value.

Exporting the WebLogic Server Domain Applications: Examples

 **Note:**

While running the export domain script on a Windows command shell, you *must* escape the path separators.

Example 18-1 Running the Export Domain Script on UNIX

In UNIX, the following example exports the domain from the path, `/Oracle_Home/user_projects/domains/base_domain`. The argument `/usr/myUserKeyFile` is the path to the encryption key file and `download/com.oracle.weblogic.management.tools.migration.jar` is the path to the export tool JAR file.

```
exportDomainForPartition.sh /Oracle_Home /Oracle_Home/user_projects/domains/
base_domain /usr/myUserKeyFile /download/
com.oracle.weblogic.management.tools.migration.jar
```

Or

```
export-domain.sh -oh /Oracle_Home -domainDir /Oracle_Home/user_projects/domains/
base_domain -keyFile /usr/myUserKeyFile
```

Example 18-2 Running the Export Domain Script on Windows

In Windows, the following example exports the domain from the path, `Oracle_Home\user_projects\domains\base_domain`. The argument `myKeyfile` is the encryption key file and `oracle.weblogic.management.tools.migration.jar` is the export tool JAR file.

```
exportDomainForPartition.cmd C:\\Oracle_Home C:\\Oracle_Home\\user_projects\\domains
\\base_domain myKeyfile C:\\com.oracle.weblogic.management.tools.migration.jar
```

Or

```
export-domain.cmd -oh C:\\Oracle_Home -domainDir C:\\Oracle_Home\\user_projects\\
\\domains\\base_domain -keyFile myKeyfile
```

Using the JSON File to Override Defaults During Import

During the operation of exporting the WebLogic Server domain applications, a JSON text file is generated and stored both as an archive file and a separate file that can be edited and modified. This file specifies default values for the partition that is created during the import operation. However, you can edit the JSON file to override these default values. For instance, the JSON file specifies a default virtual target name. If you want to create a virtual target with a different name, then you can edit the JSON file to alter the value in the `virtual-target` section.

The following example shows a sample JSON file that is generated by the export tool and illustrates both the JSON file objects and attributes, and how they can be overridden:

```
{
  "virtual-target": [{
    "name": "${PARTITION_NAME}-AdminServer-virtualTarget",
    "target": "AdminServer",
    "uri-prefix": "/${PARTITION_NAME}"
  }],
  "resource-group": [{
    "name": "${PARTITION_NAME}-AdminServer-RG",
    "target": [{
      "virtual-target": {
        "name": "${PARTITION_NAME}-AdminServer-virtualTarget"
      }
    }
  ]],
  "app-deployment": [{
    "name": "sampleStandaloneApplication1",
    "exclude-from-import": "false",
    "sub-module-targets": [
      {
        "name": "sampleAppSubdeployment1",
        "targets": "__EXISTING_VALUE__"
      }
    ]
  }],
  {
    "name": "sampleJMSApplication1",
    "exclude-from-import": "false",
    "jms-modules": [
      {
        "name": "sampleAppJmsModule1",
        "sub-module-targets": [
          {
            "name": "sampleAppJmsModuleSubdeployment1",
            "targets": "__EXISTING_VALUE__"
          }
        ]
      }
    ]
  },
  {
    "name": "sampleAppJmsModule2",
```

```

        "sub-module-targets": [
            {
                "name": "sampleAppJmsModuleSubdeployment2",
                "targets": "__EXISTING_VALUE__"
            }
        ]
    }
}
}],
"library": [{
    "name": "sampleCustomLibrary1",
    "exclude-from-import": "false"
},
{
    "name": "sampleSharedLibrary1",
    "exclude-from-import": "false",
    "source-path": "${WL_HOME}\common\deployable-libraries\
sampleSharedLibrary1.war",
    "require-exact-version": "false"
}],
"file-store": [{
    "name": "sampleFileStore1",
    "exclude-from-import": "false"
},
{
    "name": "sampleFileStore2",
    "exclude-from-import": "false"
}],
"jdbc-store": [{
    "name": "sampleJDBCStore1",
    "exclude-from-import": "false",
    "prefix-name": "__EXISTING_VALUE__"
},
{
    "name": "saf_jdbc_store_3",
    "exclude-from-import": "false",
    "prefix-name": "__EXISTING_VALUE__"
}],
"messaging-bridge": [{
    "name": "SampleJMSMsgBridge",
    "exclude-from-import": "false",
    "jms-bridge-destination": [{
        "name": "SrcBridgeDestn",
        "connection-url": "__EXISTING_VALUE__",
        "user-name": "__EXISTING_VALUE__",
        "user-password": "__EXISTING_VALUE__"
    },
    {
        "name": "TgtBridgeDestn",
        "connection-url": "__EXISTING_VALUE__",
        "user-name": "__EXISTING_VALUE__",
        "user-password": "__EXISTING_VALUE__"
    }
    ]
}],
"jms-server": [{
    "name": "sampleJMSServer1",
    "exclude-from-import": "false"
},
{
    "name": "sampleJMSServer2",
    "exclude-from-import": "false"
}

```

```

    }],
    "jdbc-system-resource": [{
        "name": "SampleDataSourceXA",
        "exclude-from-import": "false"
    }],
    "jms-system-resource": [{
        "name": "sampleJMSModule1",
        "exclude-from-import": "false",
        "sub-deployment": [{
            "name": "sampleSubDeployment1",
            "exclude-from-import": "false"
        }]
    }],
    {
        "name": "sampleJMSModule2",
        "exclude-from-import": "false",
        "sub-deployment": [{
            "name": "sampleSubDeployment2",
            "exclude-from-import": "false"
        }],
        "saf-remote-context": [{
            "name": "sample_saf_remote_ctx",
            "loginURL": "__EXISTING_VALUE__",
            "username": "__EXISTING_VALUE__",
            "password": "__EXISTING_VALUE__"
        }]
    }],
    "resource-group-template": {
        "name": "${PARTITION_NAME}-AdminServer-RGTemplate"
    }
}],
"partition": {
    "default-target": [{
        "virtual-target": {
            "name": "${PARTITION_NAME}-AdminServer-virtualTarget"
        }
    }],
    "jdbc-system-resource-override": [{
        "name": "SampleDataSourceXA",
        "url": "__EXISTING_VALUE__",
        "user": "__EXISTING_VALUE__",
        "password-encrypted": "__EXISTING_VALUE__"
    }],
    "jms-system-resource-override": [{
        "name": "sampleJMSModule1",
        "foreign-server-override": [{
            "name": "ForeignServer1",
            "foreign-destination-override": [{
                "name": "ForeignDestination1",
                "remote-jndi-name": "__EXISTING_VALUE__"
            }],
            {
                "name": "ForeignDestination2",
                "remote-jndi-name": "__EXISTING_VALUE__"
            }
        }],
        "foreign-connection-factory-override": [{
            "name": "ForeignConnectionFactory1",
            "remote-jndi-name": "__EXISTING_VALUE__",
            "username": "__EXISTING_VALUE__"
        }]
    }
}
}

```

```

    }],
    "realm": "__EXISTING_VALUE__",
    "available-target": [{
      "virtual-target": {
        "name": "${PARTITION_NAME}-AdminServer-virtualTarget"
      }
    }]
  },
  "implementation-version": "12.2.1.2"
}

```

The following table describes the objects and attributes that can be edited or modified in the JSON file.

Objects and Attributes in the JSON File that Can Be Edited	Notes
Root level object "virtual-target"	<p>By default, the import tool creates virtual targets as shown in the sample file. Oracle recommends that you manually configure virtual targets before the import operation. For information about configuring virtual targets, see Configuring Virtual Targets.</p> <p>Ensure that you remove this object and any related value to prevent the automatic creation of virtual targets during the import operation.</p>
Object "partition"	<p>This object specifies elements and values required for the creation of the partition.</p> <p>If you have already manually created virtual targets for this partition, then replace the "virtual-target" value (within "default-target" and "available-target") with the name of the existing virtual target.</p>
Object "resource-group-template"	<p>To override the default resource group template name, edit the "name" value.</p>
Object "resource-group"	<p>This attribute determines whether to exclude a particular object or resource from importing to a partition. See Migrating a WebLogic Server Domain to a Domain Partition: Limitations and Considerations for restrictions on JMS and JDBC resource targeting in partitions.</p> <p>Set this value to <code>true</code> to prevent an object or a resource from importing to a partition. The default value is <code>false</code>.</p>
Attribute "exclude-from-import"	

Objects and Attributes in the JSON File that Can Be Edited	Notes
Attribute "require-exact-version" for shared libraries.	<p>This attribute determines whether to use the shared library with a different implementation version that is available at the target or use the shared library imported from the source domain.</p> <p>Set this value to <code>true</code> if you want to deploy the library file available at the source domain to the target. The deployed library file will be located at the domain directory at <code>..<i>domainName</i>\servers\<i>servername</i>\upload\resource-group-templates\...</code></p> <p>The default value is <code>false</code>. If set to <code>false</code>, then the library available at the target location at <code>../common/deployable-libraries</code>, even if it has a different implementation version (but has the same specification version), will be deployed and the source domain library will be ignored. If the same library is not available or if the specification version is different, then the source domain library will be used.</p>

Importing an Applications Archive File to a Domain Partition

Prior to importing the domain application archive file ensure that the target WebLogic Server 12.2.1.x domain is up and running.

To create a new partition in the WebLogic Server 12.2.1.x domain and import the applications archive file to the newly created domain partition, perform the following steps using WLST commands in the online mode.

1. Connect to the target WebLogic Server domain where you want to import the domain archive file, as shown in the following example:

```
connect('user', 'password', 't3://myserver:7001')
```

2. Run the `importPartition` command using the following syntax:

```
importPartition( archiveFileName , partitionName , createRGT=true|false ,
userKeyFile )
```

The command is asynchronous and returns the MBean `ImportExportPartitionTaskMBean`. This command can be used to obtain the result of the import operation in the following manner:

```
result = importPartition( ... )
```

You can use `print result.getState()` to obtain the import result state or `print result.getError()` to see the error message in case the import operation fails. For more information about this command, see *WLST Command Reference for WebLogic Server*.

The following table describes the arguments that must be specified for the `importPartition` command.

Argument	Description
<code>archiveFileName</code>	The full path to the partition archive to import. The path must be reachable by the Administration Server. This command also looks for a file, <i>domain name-attributes.json</i> in the same directory as the domain archive. If it is found then the values in that file overrides those in the domain archive.
<code>partitionName</code>	Optional. The name to use for partition when it is created in the target domain. This must be specified only when the <code>importpartition</code> command is used to import a domain archive created by D-PCT.
<code>createRGT</code>	Optional. A flag that indicates whether to create a resource group template for all the resources in the archive file. Set to <code>true</code> if you want to create a resource group template, or set to <code>false</code> if you want all resources to be added directly to the resource group of the new partition.
<code>userKeyFile</code>	Optional. The absolute path to the user key file containing the same clear-text passphrase specified during the export operation. This file is used to decrypt attributes in the archive file.

Example 18-3 Importing an Applications Archive File to a Domain Partition: WLST Example

The following WLST command imports the `outDir/<domain>.zip` archive file to the new partition called `testPartition`. It also creates a resource group template and uses `/usr/myUserKeyFile` as the encryption key.

```
wls:/wsDomain/> importPartition( '/outDir/<domain>.zip', 'testPartition', true,
'/usr/myUserKeyFile' )
```

Monitoring the Export and Import Operation using Reports

You can monitor and analyze the export and import operation using the reports generated by D-PCT. The export and import reports provide useful information when you migrate resources from the source domain to the target domain partition using D-PCT.

The export report gets stored in the directory where the domain archive is created under the D-PCT installation directory, in the format *domain name-ExportLogReport.html*. It contains information such as, the domain location, the source WebLogic Server version, along with a list of all resources that are supported or not supported for export by the tool. You can use the export report to view the list of servers and resources that were successfully exported or the ones that failed to export. You can also view the list of resources that were untargeted and not exported, or the resources that were targeted (to more than one server or cluster) but were not exported.

The import report gets stored in the domain directory in the format, *PartitionName_MigrationImportReport.html*. This report contains information about the virtual targets that get created during the import operation, along with the resource group and their targeting information. You can use this report to view the list of resources that were successfully imported to the domain partition. It also lists the resources that were excluded by the user, or the ones which were ignored by the tool because they were not compatible in a multitenant environment.

Migrating a WebLogic Server Domain to a Domain Partition: Limitations and Considerations

Certain scenarios are not supported by the Domain to Partition Conversion Tool.

- The domain upgrade of libraries and resources to a new release is not supported at import. Administrators and users are responsible for possible necessary domain upgrades from the source domain before exporting the archive file.
- Neither the application runtime state nor the application specific runtime configuration is exported to the archive file. For example, the JMS messages in queues or users in an embedded LDAP realm are not exported.
- The remote clients compiled with WebLogic Server versions prior to 12.2.1 are not able to look up JNDI resources deployed on a partition. They need to be recompiled with the WebLogic Server releases 12.2.1 or later.
- JDBC does not support the use of the Logging Last Resource (LLR) feature in WebLogic Server Multitenant. Data sources with this option need to be converted to use a substitute setting that may or may not be an adequate replacement.

Considerations and Limitations for Importing a Domain with JMS Resources

This section lists the limitations and other behaviors that you must consider before exporting a domain containing one or more JMS resources.

- If a subdeployment within a JMS system module is targeted to one or more cluster(s) or migratable target(s) or WebLogic Server instances, and if it is referred by a uniform distributed destination, then you must modify the subdeployment targeting to the respective JMS servers that are targeted to the cluster or migratable target or WebLogic Server.
- Uniform Distributed Destinations and SAF Imported Destinations using default targeting in the source domain will be left untargeted after successful import. If a uniform distributed destination or a SAF imported destination or a Foreign Server is default targeted, then you must modify them to use subdeployment targeting by targeting the resources to appropriate JMS servers or SAF Agents.
- Ensure that a single JMS server does not host conflicting JMS Resources, such as hosting a standalone destination *and* a uniform distributed destination. During the export operation, if D-PCT finds a JMS server hosting one of the conflicting resources, then the corresponding JMS system module is omitted from export. This is because each of these resources require a JMS server that references a persistence store with varying distribution policies. See [Configuring JMS Servers](#).
- Weighted distributed destinations are deprecated and are not supported by JMS in a multitenant environment. Therefore, D-PCT ignores weighted distributed

destinations. Oracle recommends that you convert any weighted distributed destination resource to its equivalent uniform distributed destination resource. For information about configuring the different types of distributed destination resources, see *Configuring Distributed Destination Resources in Administering JMS Resources for Oracle WebLogic Server*.

- At the domain level, JMS resources can be targeted to multiple WebLogic servers or clusters. However in a partitioned environment, the JMS resources are not supported if the resource group is targeted to multiple virtual targets. Therefore, D-PCT ignores the application deployments and all other resources targeted to more than one WebLogic Server instance or cluster or migratable targets in the source domain.
- JMS does not support the `Replicated` forwarding policy for a uniform distributed topic in a partitioned environment. Therefore, in such cases, D-PCT converts the forwarding policy to `Partitioned`. For information about the uniform distributed topic message forwarding policy, see *Configuring Partitioned Distributed Topics in Administering JMS Resources for Oracle WebLogic Server*.
- If there are more than one JMS servers, each targeted to a migratable target or a WebLogic Server member of the same cluster, then the import operation consolidates all JMS servers to a single JMS server configured with a store having the distribution policy `Distributed`. The other JMS servers are ignored during the import operation.
- In a partitioned environment, JMS allows a subdeployment to have only one JMS server or one SAF agent as its target. If the source domain has a subdeployment targeted to more than one JMS server or SAF Agent, then they would be sorted alphabetically and the first one would be chosen as the target of the subdeployment. Default targeted Connection Factories are migrated as it is and are referenced to the target of the resource group under which the JMS system module is created.
- In a partitioned environment, only the JMS server referencing the persistence store with distribution policy `Singleton` can host standalone destinations. The source domains supported by D-PCT do not support distribution policy since the concept of distribution policy is introduced recently in WebLogic Server 12.2.1. To workaround this limitation set by JMS, D-PCT automatically sets the distribution policy of the persistence store referenced by JMS Servers hosting standalone destinations.
- At the domain level configuration, JMS allows multiple messaging bridges to use the same bridge destination as the source or the target destination. However, the new validation introduced by JMS requires that when the messaging bridge and bridge destinations are configured in a partitioned environment, the bridge destination can be used either as a source or the target destination by only one messaging bridge. Keeping this in mind, D-PCT takes the following approach during migration to a partition:
 - A bridge destination from the source domain is created in the target partition *only* when a message bridge references that bridge destination.
 - When D-PCT finds a bridge destination that is already used by a messaging bridge as a source or a target destination, the name of the bridge destination is internally modified to *original bridge destination name_the name of the messaging bridge that references this bridge destination*.

 **Note:**

D-PCT allows you to exclude one or more application deployments or resources from importing if those resources are not supported by the tool. You can exclude such resources from importing by setting the `exclude-from-import` attribute in the JSON file to `true`. Such resources can, then, be manually created within the partition as needed.

19

Configuring Partition Concurrent Managed Objects

The concurrency utilities for Java EE 1.0, Java Specification Request (JSR) 236, implements a standard API for providing asynchronous capabilities to Java EE application components, such as servlets and EJBs. In WebLogic Server, concurrent managed objects (CMOs) provide concurrency capabilities to Java EE applications. You can configure concurrent managed objects and then make them available for use by application components. Learn how to configure partition-level *concurrent managed object (CMO) templates* and *concurrent constraints* in Oracle WebLogic Server Multitenant (MT).

Note:

WebLogic Server Multitenant domain partitions, resource groups, resource group templates, virtual targets, and Resource Consumption Management are deprecated in WebLogic Server 12.2.1.4.0 and will be removed in the next release.

This chapter includes the following sections:

- [Configuring Partition Concurrent Managed Templates: Overview](#)
- [Configuring Partition CMO Templates: Main Steps](#)
- [Configuring Partition Concurrent Constraints: Main Steps](#)
- [Configuring Partition Concurrent Managed Objects: Related Tasks and Links](#)

Configuring Partition Concurrent Managed Templates: Overview

In addition to providing support for domain- and server-level CMOs, WebLogic Server also provides partition-level CMO templates to perform partition-scoped task execution.

This document discusses CMO templates only as they apply to partitions.

Concurrent Managed Object Components

The primary components of the concurrency utilities are:

- **ManagedExecutorService (MES):** Used by applications to execute submitted tasks asynchronously. Tasks are executed on threads that are started and managed by the container. The context of the container is propagated to the thread executing the task.

- **ManagedScheduledExecutorService (MSES):** Used by applications to execute submitted tasks asynchronously at specific times. Tasks are executed on threads that are started and managed by the container. The context of the container is propagated to the thread executing the task.
- **ManagedThreadFactory (MTF):** Used by applications to create managed threads. The threads are started and managed by the container. The context of the container is propagated to the thread executing the task.
- **ContextService:** Used to create dynamic proxy objects that capture the context of a container and enable applications to run within that context at a later time or be submitted to a Managed Executor Service. The context of the container is propagated to the thread executing the task.

For more detailed information, see [Concurrency Utilities for Java EE](#) in *The Java EE 7 Tutorial*. Also see the Java Specification Request 236: Concurrency Utilities for Java EE 1.0 (<http://jcp.org/en/jsr/detail?id=236>).

Partition-Level Asynchronous Task Management by CMOs

In WebLogic Server, asynchronous task management is provided by four types of CMOs, as summarized in [Table 19-1](#).

Table 19-1 CMOs That Provide Asynchronous Task Management

Managed Object	Context Propagation	Self-Tuning	Thread Interruption While Shutting Down	Limit of Concurrent Long-Running New Threads
Managed Executor Service (MES)	Contexts are propagated based on configuration.	Only short-running tasks are dispatched to the single self-tuning thread pool by a specified Work Manager.	When the Work Manager is shutting down, all the unfinished tasks will be canceled.	The maximum number of long-running threads created by MES/MSES can be configured to avoid excessive number of these threads making negative effects on server.
Managed Scheduled Executor Service (MSES)	Contexts are propagated based on configuration.	Same behavior as MES.	Same behavior as MES.	Same behavior as MES.
Context Service	Contexts are propagated based on configuration.	NA	NA	NA
Managed Thread Factory (MTF)	Contexts are propagated based on configuration.	Threads returned by the <code>newThread()</code> method are not from the single self-tuning thread pool and will not be put into the thread pool when the task is finished.	Threads created by the <code>newThread()</code> method will be interrupted when the MTF is shutting down.	The maximum number of new threads created by MTF can be configured to avoid excessive number of these threads making negative effects on server.

There are three types of JSR236 CMOs in WebLogic Server, each one characterized by its scope and how it is defined and used.

- Default Java EE CMOs: Required by the Java EE standard that default resources be made available to applications, and defines specific JNDI names for these default resources.
- Customized CMOs in configuration files: Can be defined at the application and module level or referenced from an application component environment that is bound to JNDI.
- Global CMO templates: Can be defined globally as templates in the domain, server, and partition level configuration by using the WebLogic Server Administration Console and configuration MBeans.

 **Note:**

This document discusses global CMO templates only as they apply to domain partitions. For detailed conceptual information about default Java EE CMOs and customized CMOs in configuration files, see *Configuring Concurrent Managed Objects in Administering Server Environments for Oracle WebLogic Server*.

Similar to Work Managers, global CMO templates can be defined at the domain, server, and partition level using the WebLogic Server Administration Console or configuration MBeans. For more information about configuring CMOs at the partition level, see [Configuring Partition CMO Templates: Main Steps](#).

Partition-Level MES Template Configuration Elements

[Table 19-2](#) defines the configuration elements for a partition-level MES template.

Table 19-2 Managed Executor Service Configuration Elements

Name	Description	Required	Default Value	Range
name	<p>Name of the MES.</p> <ul style="list-style-type: none"> The name should not be <code>DefaultManagedExecutorService</code>, otherwise the server cannot be started normally. Partition-level MES with the same name can NOT be configured in the same partition, otherwise the server cannot be started normally. An <code>ManagedExecutorService</code> can have the same name as other types of managed objects such as a <code>ContextService</code> in any partition with no relationship between them. A partition-level MES and a JSR236 MES can have the same name with no relationship between them. If a partition-level MES with the same name is defined in both <code><resource-group-template></code> and one partition's <code><resource-group></code>, then the <code><resource-group></code> definition will override <code><resource-group-template></code> definition in that partition. 	Yes	NA	An arbitrary non-empty string.
dispatch-policy	Name of the Work Manager. The partition-level Work Manager will be picked by this name. If the Work Manager does not exist, use the partition-level default Work Manager.	No	Default Work Manager	NA
max-concurrent-long-running-requests)	Maximum number of concurrent long running tasks.	No	10	[0-65534]. When out of range, the default value will be used.

Table 19-2 (Cont.) Managed Executor Service Configuration Elements

Name	Description	Required	Default Value	Range
long-running-priority	An integer that specifies the long-running daemon thread's priority. If specified, then all long-running threads will be affected.	No	Thread.NORM_PRIORITY	1-10 Range between Thread.MIN_PRIORITY, and Thread.MAX_PRIORITY. When out of range, the default value will be used.

Partition-Level MSES Template Configuration Elements

[Table 19-3](#) defines the configuration elements for a partition-level MSES template, which are similar to a partition-level MES template.

Table 19-3 Managed Scheduled Executor Service Configuration Elements

Name	Description	Required	Default Value	Range
name	The name of the MSES. For naming convention rules, see Table 19-2 .	Yes	NA	An arbitrary non-empty string.
dispatch-policy	The name of the Work Manager. For Work Manager usage rules, see Table 19-2 .	No	Default Work Manager	NA
max-concurrent-long-running-requests	Maximum number of concurrent long running tasks.	No	10	[0-65534]. When out of range, the default value is used.
long-running-priority	An integer that specifies the long-running daemon thread's priority. If specified, then all long-running threads will be affected.	No	5 Thread.NORM_PRIORITY	1-1 Range between Thread.MIN_PRIORITY and Thread.MAX_PRIORITY. When out of range, the default value is used.

Partition-Level MTF Template Configuration Elements

[Table 19-4](#) defines the configuration elements for a partition-level MTF template.

Table 19-4 Managed Thread Factory Configuration Elements

Name	Description	Required	Default Value	Range
name	The name of the MTF. For naming convention rules, see Table 19-2 .	Yes	NA	An arbitrary non-empty string.
priority	The priority to assign to the thread. (The higher the number, the higher the priority.)	No	5 Thread.NORM_PRIORITY	1-10 Range between Thread.MIN_PRIORITY and Thread.MAX_PRIORITY. When out of range, the default value is used.
max-concurrent-new-threads	The maximum number of threads created by the MTF that are still executing the <code>run()</code> method of the tasks.	No	10	[0-65534] When out of range, the default value is used.

[Table 19-5](#) defines the configuration elements for a partition-level MTF.

Table 19-5 Managed Thread Factory Configuration Elements

Name	Description	Required	Default Value	Range
name	The name of the MTF. For naming convention rules, see Table 19-2 .	Yes	NA	An arbitrary non-empty string.
priority	The priority to assign to the thread. (The higher the number, the higher the priority.)	No	5 Thread.NORM_PRIORITY	1-10 Range between Thread.MIN_PRIORITY and Thread.MAX_PRIORITY. When out of range, the default value is used.
max-concurrent-new-threads	The maximum number of threads created by the MTF that are still executing the <code>run()</code> method of the tasks.	No	10	[0-65534] When out of range, the default value is used.

Partition-Level CMO Constraints for Long-Running Threads

Long-running tasks submitted to MES and MSES and the calling of `newThread()` method of MTF need to create new threads that will not be managed as a part of the self-tuning thread pool. Because an excessive number of running threads can have a

negative effect on server performance and stability, configurations are provided to specify the maximum number of running threads that are created by concurrency utilities API.

Setting Limits for Maximum Concurrent Long-Running Requests

The limit of concurrent long-running requests submitted to MES and MSES can be specified in partitions. All levels of configurations are independent and the maximum of the concurrent long-running requests cannot exceed any of them.

[Table 19-6](#) summarizes the limit of concurrent long-running requests with the `max-concurrent-long-running-requests` element that can be defined in the deployment descriptors.

Table 19-6 Limit of Concurrent Long-Running Requests in Partitions

Scope	Deployment Descriptor	Description	<max-concurrent-long-running-requests> Element Details
Partition	In <code>config.xml</code> : As the subelement of <code><domain><partition></code> for common partition. As the subelement of <code><domain></code> for global runtime.	Limit of concurrent long-running requests specified for that partition in that server.	Optional Range: [0-65534]. When out of range, the default value will be used. Default value: 50

When the specified limit is exceeded, the MES or MSES takes the following actions for new long-running tasks submitted to them:

- The `java.util.concurrent.RejectedExecutionException` is thrown when calling the task submission API.
- If the user registered the task with the `ManagedTaskListener`, then this listener is not notified because the submit method failed.

Note that the preceding rule is not applied for the `invokeAll()` and `invokeAny()` methods. If any of the tasks submitted by these methods is rejected by the specified limit, then the following events occur:

- The user-registered managed task listener's `taskSubmitted()` method is called.
- The user-registered managed task listener's `taskDone()` method is called and the throwableParam is `javax.enterprise.concurrent.AbortedException`.
- Other submitted tasks are not affected.
- The method does *not* throw the exception `RejectedExecutionException`.

The following example demonstrates how the value specified for the `max-concurrent-long-running-requests` element in the `config.xml` file can affect the maximum number of long-running requests.

```
<domain>
  <server>
    <name>myserver</server>
    <max-concurrent-long-running-requests>50</max-concurrent-long-running-requests> (place 1)
```

```

    </server>
    <max-concurrent-long-running-requests>10</max-concurrent-long-running-requests>
(place 2)
    <partition>
        <name>mypartition</name>
        <max-concurrent-long-running-requests>5</max-concurrent-long-running-requests>
(place 3)

    </partition>
    <server-template>
        <name>mytemplate</name>
        <max-concurrent-long-running-requests>50</max-concurrent-long-running-
requests> (place 4)
    </server-template>
</domain>

```

- place 1: Affects the MES and MSES defined in the server instance *myserver*. All the MES and MSES running in that server instance can create only a maximum of 50 long-running-requests in total.
- place 2: Only affects MES and MSES defined in the domain. It does not affect partition-level MES and MSES. All the MES and MSES running in the domain can create a maximum of 10 long-running requests in total.
- place 3: Only affects MES and MSES defined in *mypartition*. All the MES and MSES running in partition *mypartition* can create maximally 5 long-running requests in total.
- place 4: Affects MES and MSES defined in the server instances that apply to the template *mytemplate*. All the MES and MSES running in that server instance can create only a maximum of 50 long-running-requests in total.

The following example demonstrates how the value specified for the `max-concurrent-long-running-requests` element in the `config.xml` file can affect the maximum number of long-running requests.

In the following case, none of the limits is exceeded and the prior actions are not taken:

- Assume 120 long-running tasks were submitted to `managed-executor-service1`, 115 of them were finished, 5 of them are being executed, if one more long-running task is submitted to `managed-executor-service1`, then it is executed because no limit is exceeded.

In the following cases, one of the limits is exceeded and the prior actions are taken:

- Assume 10 long-running tasks are being executed by `managed-scheduled-executor-service1`, if one more long-running task is submitted to `managed-scheduled-executor-service1`, then the limit of `managed-scheduled-executor-service1` is exceeded.
- Assume 10 long-running tasks are being executed by `application1`, and 40 are being executed by `application2`; if one more long-running task is submitted to `application1` or `application2`, then the limit of `partition1` is exceeded.
- Assume 10 long-running tasks are being executed by `application1`, 30 are being executed by `application2`, and 60 are being executed by `partition2`; if one more long-running task is submitted to `application1`, or `application2`, or `partition2`, then the limit of `server1` is exceeded.

```
server1(100)
  |---partition1(50)
    |---application1
      |---managed-scheduled-executor-service1(not specified)
      |---module1
        |---managed-executor-service1(20)
        |---managed-scheduled-executor-service2(not specified)
      |---application2
    |---partition2(70)
```

Setting Limits for Maximum Concurrent New Threads

The limit of concurrent new running threads created by calling the `newThread()` method of the MTF can be specified in a managed object, domain, and server level. All levels of configurations are independent and the maximum of the concurrent new running threads cannot exceed any of them.

Note that the meaning of **running thread** is a thread that has been created by the MTF that has not finished its `run()` method.

[Table 19-7](#) summarizes the limit of concurrent new running threads with an element `<max-concurrent-new-threads>` that can be defined in the deployment descriptors.

Table 19-7 Limit of Concurrent New Running Threads

Scope	Deployment Descriptor	Description	<code><max-concurrent-long-running-requests></code> Element Details
Partition	In <code>config.xml</code> : As the subelement of <code><domain><partition></code> for common partition. As the subelement of <code><domain></code> for global runtime.	Limit of concurrent new running threads specified for that partition in that server.	Optional Range: [0-65534], when out of range, the default value is used. Default value: 50

When the specified limit is exceeded, calls to the `newThread()` method of the MTF will return `null` to be consistent with the `ThreadFactory.newThread` Javadoc.

To see a sample of using `max-concurrent-new-threads`, see [#unique_379/unique_379_Connect_42_CHDJIBCE](#).

Configuring Partition CMO Templates: Main Steps

You can define global CMOs as templates in the partition's configuration by using the WebLogic Server Administration Console and configuration MBeans. CMO templates can be assigned to any application, or application component in the partition.

 **Note:**

Partitioned domains for multitenancy are only available in WebLogic Server Premium Edition. See About WebLogic Server MT.

You can define three types of CMO templates in a partition:

- Managed Executor Service
- Managed Scheduled Executor Service
- Managed Thread Factory

For example, if you define a `managed-executor-service-template`, a unique MES instance is created for each application deployed in the partition.

Using the Administration Console to Configure a Partition-Scoped CMO Template

CMO templates can be configured and scoped to a partition using the WebLogic Server Administration Console.

1. In the **Domain Structure** tree, expand **Environment** and click **Concurrent Templates**.
2. Click **New** and select one of the following template options:
 - **Managed Executor Service Template**
 - **Managed Scheduled Executor Service Template**
 - **Managed Thread Factory Template**
3. On the **Create New Template** page, enter the template properties as required. The properties vary depending on which type of concurrent template you are creating.
 - See [Partition-Level MES Template Configuration Elements](#).
 - See [Partition-Level MSES Template Configuration Elements](#)
 - See [Partition-Level MTF Template Configuration Elements](#).
4. Click **Next**.
5. Select the partition that you want to target the concurrent template to.

Only applications that have been deployed to the selected partition can use this concurrent template.
6. Click **Finish**. The Summary of Concurrent Templates page displays and the new partition-scoped concurrent template is listed.
7. Repeat these steps to create other concurrent templates as necessary.

For more detailed information about configuring CMO templates for a domain and server scope, see Global CMO Templates in *Administering Server Environments for Oracle WebLogic Server*.

Using MBeans to Configure CMO Templates

CMO templates can be configured using the following configuration MBeans under `PartitionMBean`:

- [ManagedExecutorServiceTemplateMBean](#)
- [ManagedScheduledExecutorServiceTemplateMBean](#)
- [ManagedThreadFactoryTemplateMBean](#)

See the [PartitionMBean](#) section in the *MBean Reference for Oracle WebLogic Server*.

Configuring Partition Concurrent Constraints: Main Steps

You can also define constraints for a partition scope using the WebLogic Server Administration Console and configuration MBeans.

- [Using the Administration Console to Configure Concurrent Constraints for a Partition](#)
- [Using MBeans to Configure Concurrent Constraints](#)

Using the Administration Console to Configure Concurrent Constraints for a Partition

Concurrent constraints can be configured per partition using the WebLogic Server Administration Console.

1. In the **Domain Structure** tree, click **Domain Partitions**.
2. In the **Summary of Domain Partitions** table, select the partition that you want to configure.
3. In the **Settings for *partition-name*** page, open the **Configuration > Concurrency** page for the partition
4. Specify a value for any or all of the concurrent constraint options:
 - **Max Concurrent Long Running Requests:** The maximum number of concurrent long-running requests that can be submitted to all of the Managed Executor Services or Managed Scheduled Executor Services in the partition scope.
See [Setting Limits for Maximum Concurrent Long-Running Requests](#).
 - **Max Concurrent New Threads:** The maximum number of concurrent new threads that can be created by all of the Managed Thread Factories in the partition scope.
See [Setting Limits for Maximum Concurrent New Threads](#).
5. Click **Save**.

Using MBeans to Configure Concurrent Constraints

Concurrent constraints can be configured per partition using the following methods under the `PartitionMBean`:

- `maxConcurrentLongRunningRequests()` The maximum number of concurrent long-running requests that can be submitted to all of the Managed Executor Services or Managed Scheduled Executor Services in the partition scope.
See [Setting Limits for Maximum Concurrent Long-Running Requests](#).
- `maxConcurrentNewThreads()` The maximum number of concurrent new threads that can be created by all of the Managed Thread Factories in the partition scope.
See [Setting Limits for Maximum Concurrent New Threads](#).

For more information about using WebLogic Server MBeans, see *Accessing WebLogic Server MBeans with JMX in Developing Custom Management Utilities Using JMX for Oracle WebLogic Server*.

Monitoring Partition-level CMO Templates

Concurrent constraints can be queried directly, through MBeans, and by using the WebLogic Server Administration Console.

- [Using the Administration Console to Monitor CMO Templates](#)
- [Using MBeans to Monitor CMO Templates](#)

Using the Administration Console to Monitor CMO Templates

CMO templates can be monitored using the Administration Console.

To monitor the threads created by CMOs in a partition:

1. In the **Domain Structure** tree, click **Domain Partitions**.
2. In the **Summary of Domain Partitions** table, select the partition you want to monitor.
3. In the **Settings for *partition-name*** page, select the **Monitoring > Concurrent Templates** tab.

The table provides configuration information for CMO templates that have been targeted to the partition.

Using MBeans to Monitor CMO Templates

A partition's CMO templates can be monitored using the `ConcurrentManagedObjectsRuntimeMBean`, which can be accessed from the following MBean attribute:

- `PartitionRuntimeMBean.ConcurrentManagedObjectsRuntime` – Provides configuration information for CMO templates that have been targeted to the partition.

See the [ConcurrentManagedObjectsRuntimeMBean](#) in the *MBean Reference for Oracle WebLogic Server*

For more information about using WebLogic Server MBeans, see *Accessing WebLogic Server MBeans with JMX in Developing Custom Management Utilities Using JMX for Oracle WebLogic Server*.

Configuring Partition Concurrent Managed Objects: Related Tasks and Links

- *Configuring Concurrent Managed Objects in Administering Server Environments for Oracle WebLogic Server*.
- For more information about CMO-related configuration and runtime MBeans, see the [ManagedExecutorServiceTemplateMBean](#), [ManagedScheduledExecutorServiceTemplateMBean](#),

[ManagedThreadFactoryTemplateMBean](#), and the [PartitionMBean](#) in the *MBean Reference for Oracle WebLogic Server*.

- For more information about using WLST commands, see WebLogic Server WLST Online and Offline Command Reference in the *WLST Command Reference for WebLogic Server*.
- Sample Applications and Code Examples in *Understanding Oracle WebLogic Server*.
- [Concurrency Utilities for Java EE](#) in *The Java EE 7 Tutorial*.
- The Java Specification Request 236: Concurrency Utilities for Java EE 1.0 (<http://jcp.org/en/jsr/detail?id=236>).

Configuring Partition Batch Job Runtime

Batch jobs are tasks that can be executed without user interaction and are best suited for non-interactive, bulk-oriented, long-running tasks that are resource intensive. They can execute sequentially or in parallel, and may be initiated dynamically or through scheduling. WebLogic Server supports batch job runtimes that are scoped to partitions.

 **Note:**

WebLogic Server Multitenant domain partitions, resource groups, resource group templates, virtual targets, and Resource Consumption Management are deprecated in WebLogic Server 12.2.1.4.0 and will be removed in the next release.

This chapter includes the following sections:

- [Configuring Partition Batch Job Runtime: Overview](#)
- [Configuring Partition Batch Job Runtime: Main Steps](#)
- [Querying the Batch Job Runtime](#)
- [Configuring Partition Batch Job Runtime: Related Tasks and Links](#)

Configuring Partition Batch Job Runtime: Overview

Using the batch job runtime in partitions requires the configuration of a dedicated data source for each partition to access the job repository tables for batch jobs. When a Java EE component deployed to the partition submits a batch job, the batch job runtime updates the job repository tables using this data source.

Optionally, you can configure a partition batch job runtime to use an application-scoped Managed Executor Service (MES) by configuring MES templates at the domain level, but which are scoped to partitions. The MES processes the jobs asynchronously and the job repository data source stores the status of current and past jobs. However, if no MES template is targeted to the partition, the batch job runtime will use the MES that is bound to the default JNDI name of `java:comp/DefaultManagedExecutorService` (as required by the Java EE 7 specification).

WebLogic Server also supports the use of multiple batch runtime instances within the scope of a partition. However, there are important considerations to note with regards to the deployment of a batch application that is replicated in multiple Managed Server instances, in either clustered or nonclustered environments, and also with regards to load balancing and the ability to restart failed batch jobs. See *Use of Multiple Batch Runtime Instances* in *Administering Server Environments for Oracle WebLogic Server*, and *Batch Applications* in *Administering Clusters for Oracle WebLogic Server*.

For more information about configuring and managing batch job runtime for a domain scope, such as the steps for creating the job repository tables, and important considerations for the use of multiple batch runtime instances, see *Configuring the Batch Job Runtime in Administering Server Environments for Oracle WebLogic Server*.

For more information about batch jobs, batch processing, and the batch processing framework, see [Batch Processing](#) in *The Java EE 7 Tutorial*. Also see the Java Specification Request 352: Batch Applications for the Java Platform (<http://jcp.org/en/jsr/detail?id=352>). The specification defines the programming model for batch applications and the runtime for scheduling and executing batch jobs.

Configuring Partition Batch Job Runtime: Main Steps

To configure a batch job runtime in a partition, you must complete certain steps.

- If necessary, create the job repository tables to store the batch jobs.
- Configure a JDBC data source for the partition.
- Optionally, create the MES template and target it to the partition.
- Configure the partition batch job runtime using either the WebLogic Server Administration Console or WLST.

Prerequisites

To configure a batch job runtime in a partition, the following prerequisites must be completed at the domain level:

- **Job repository tables must already exist.** The batch job runtime does not create job repository tables automatically. The database administrator must create the tables prior to activating the partition.

For information about configuring job repository tables, see *Configuring the Batch Job Runtime in Administering Server Environments for Oracle WebLogic Server*.

For information about the databases supported for containing the job repository tables, see the [Oracle Fusion Middleware Supported System Configurations](#) page on the Oracle Technology Network.

- **Configure a JDBC data source for the partition.** Each partition must have a dedicated JDBC data source created for batch jobs. The batch data source's JNDI name must be set using the `PartitionMBean.setBatchJobsDataSourceJndiName`. When a Java EE component submits a batch job, the batch job runtime updates the job repository tables using this data source, which is obtained by looking up the data source's JNDI name.

For more information about configuring a JDBC data source, see *Creating a JDBC Data Source in Administering JDBC Data Sources for Oracle WebLogic Server*.

- **Optionally, create the MES template.** For optimum performance, you can configure the batch job runtime in a partition to use an application-scoped MES by configuring MES templates at the domain level, but which are scoped to partitions. This way, a new MES instance is created for each MES template, which then runs batch jobs that are submitted for Java EE components that are deployed to the partition.

However, if no name is set, then the batch job runtime uses the default Java EE MES that is bound to the default JNDI name of: `java:comp/DefaultManagedExecutorService`.

For information about configuring the MES template, see Global CMO Templates in *Administering Server Environments for Oracle WebLogic Server*.

Configuring Partition Batch Job Runtime Using the Administration Console

You can configure a dedicated batch job runtime per partition using the WebLogic Server Administration Console. In the **Settings for *partition-name*** page, go to the **Configuration > General** page for the partition and complete these configuration fields:

- **Executor Service Template:** The name of the application-scoped MES instance that is used to run batch jobs that are submitted from applications deployed to the partition.

The MES template by the same name must exist and be scoped to the partition when a batch job is submitted to the partition. If no name is set, then the batch job runtime uses the default Java EE MES that is bound to the default JNDI name of: `java:comp/DefaultManagedExecutorService`.

- **Data Source JNDI Name:** The JNDI name of the batch job runtime's data source, which is used to store data for batch jobs submitted from applications deployed to the partition. When a Java EE component submits a batch job, the batch job runtime updates the batch job repository tables using this data source, which is obtained by looking up the data source's JNDI name.

Configuring Partition Batch Job Runtime: WLST Example

You can configure a dedicated batch job runtime per partition using WLST.

You can use WLST with the `BatchRuntimeConfigMBean` and `PartitionMBean` to configure the batch job runtime to use a specific database for the job repository:

```
def update_partition_batch_config(partitionName, jndiName, schemaName):
    connect('admin','passwd')
    edit()
    startEdit()
    cd('/Partitions/' + <partitionName>)
    cmo.setDataSourceJndiName(jndiName)
    cd('/BatchConfig/')
    cmo.setSchemaName(schemaName)
    save()
    activate()
```

In this example, if the administrator has created a data source with the JNDI name `jdbc/batchDS`, then calling `update_partition_batch_config('<partitionName>', 'jdbc/batchDS', 'BATCH')` configures the batch job runtime to store all the job repository tables in the schema `'BATCH'` in the database that is pointed to by the data source that is bound to the `jndiName: 'jdbc/batchDS'`.

You can use WLST to configure the batch job runtime to use a specific MES for batch job execution. However, you must first create the MES template at the domain level and the name of the MES must be provided to the `PartitionMBean`.

```
connect('admin','passwd')
edit()
startEdit()
cd('/Partitions/' + <partitionName>)
cmo.setBatchJobsExecutorServiceName('mesName')
save()
activate()
```

`mesName` is the name of the MES template that has already been created (and targeted) to this partition.

You can configure the batch job runtime to use different MESs using the `getBatchJobsManagedExecutorServiceName()` method in the `PartitionMBean`. However, the MES template by the same name must exist and be scoped to the partition when a batch job is submitted.

See the [BatchConfigMBean](#) and the [PartitionMBean](#) in the *MBean Reference for Oracle WebLogic Server*.

For more information about using WLST commands, see WebLogic Server WLST Online and Offline Command Reference in the *WLST Command Reference for WebLogic Server*.

Querying the Batch Job Runtime

You can query the batch job runtime's job repository table per partition scope using the WebLogic Server Administration Console and Runtime MBeans.

- [Using the Administration Console to Query the Batch Job Runtime](#)
- [Using Runtime MBeans to Query the Batch Job Runtime](#)



Note:

Ensure that the database that contains the batch job repository is running. For example, the default Derby database is not automatically started when you start WebLogic Server using the `java weblogic.Server` command. If your database is not running, an exception will be thrown when you submit a batch job or when you access the `BatchJobRepositoryRuntimeMBean`, either through WLST or the WebLogic Server Administration Console.

Using the Administration Console to Query the Batch Job Runtime

You can query a job repository table using the WebLogic Server Administration Console to obtain details about batch jobs in a partition.

Get Details of All Batch Jobs

In the **Settings for *partition-name*** page, go to the **Monitoring > Batch Jobs** page to view details about all the jobs submitted by applications deployed to the partition. See [Table 20-1](#).

Table 20-1 All Batch Jobs

Element Name	Description
Job Name	The name of the batch job.
Application Name	The name of the application that submitted the batch job.
Instance ID	The instance ID.
Execution ID	The execution ID.
Batch Status	The batch status of this job.
Start Time	The start time of the job.
End Time	The completion time of the job.
Exit Status	The exit status of the job.

Get Details About a Job's Execution

You can view step execution details about a job by selecting it and clicking **View**. See [Table 20-2](#).

Table 20-2 Job Executions Details

Element Name	Description
Job Name	The name of the batch job.
Instance ID	The instance ID.
Execution ID	The execution ID.
Batch Status	The batch status of this job.
Start Time	The start time of the job.
End Time	The completion time of the job.
Exit Status	The exit status of the job.

Get Details About a Job's Step Execution

You can view metrics about each step in a job execution by selecting it and clicking **View**. See [Table 20-3](#).

Table 20-3 Step Executions Details

Element Name	Description
Step Name	The name of the batch job step.
Step ID	The step ID.
Execution ID	The execution ID.
Batch Status	The batch status of this job.
Start Time	The start time of the job.
End Time	The completion time of the job.
Exit Status	The exit status of the job.

Using Runtime MBeans to Query the Batch Job Runtime

You can query the job repository table using WLST using the `BatchJobRepositoryRuntimeMBean` to obtain details about batch jobs in a partition.

See [BatchJobRepositoryRuntimeMBean](#) in the *MBean Reference for Oracle WebLogic Server*.

Get Details of All Batch Jobs Using `getJobDetails`

The `getJobDetails()` attribute returns details about all the jobs submitted by applications deployed to the domain. Each entry in the collection contains an array of the elements in [Table 20-4](#).

Table 20-4 Elements in `getJobDetails()` Attribute

Element Name	Description
JOB_NAME	The name of the batch job.
APP_NAME	The name of the application that submitted the batch job (String).
INSTANCE_ID	The instance ID (long).
EXECUTION_ID	The execution ID (long).
BATCH_STATUS	The batch status of this job (String).
START_TIME	The start time of the job (java.util.Date).
END_TIME	The completion time of the job (java.util.Date).
EXIT_STATUS	The exit status of the job (String).

The following example of a WLST script uses the `getJobDetails()` command to print a list of all batch jobs deployed in a domain:

```
connect('admin', 'admin123')
domainRuntime()
cd('BatchJobRepositoryRuntime')
cd('myserver')
```

```

executions=cmo.getJobDetails()
print "JobName AppName InstanceID ExecutionID BatchStatus StartTime EndTime ExitStatus"
print e[0], " ", e[1], " ", e[2], " ", e[3], " ", e[4], " ", e[5], " ", e[6], " ", e[7]

```

This is sample output after running `getJobDetails()`:

```

JobName  AppName InstanceID ExecutionID BatchStatus      StartTime
EndTime                               ExitStatus
PayrollJob  lab1  9          9          COMPLETED  Fri Apr 24 10:11:00 PDT 2015  Fri Apr 24
10:11:01 PDT 2015  COMPLETED
PayrollJob  lab1  8          8          COMPLETED  Fri Apr 24 10:11:00 PDT 2015  Fri Apr 24
10:11:01 PDT 2015  COMPLETED
PayrollJob  lab1  7          7          COMPLETED  Fri Apr 24 10:11:00 PDT 2015  Fri Apr 24
10:11:01 PDT 2015  COMPLETED
PayrollJob  lab1  6          6          COMPLETED  Fri Apr 24 10:11:00 PDT 2015  Fri Apr 24
10:11:01 PDT 2015  COMPLETED
PayrollJob  lab1  5          5          COMPLETED  Fri Apr 24 10:10:57 PDT 2015  Fri Apr 24
10:10:58 PDT 2015  COMPLETED
PayrollJob  lab1  4          4          COMPLETED  Fri Apr 24 10:10:56 PDT 2015  Fri Apr 24
10:10:56 PDT 2015  COMPLETED
PayrollJob  lab1  3          3          COMPLETED  Mon Apr 20 11:32:12 PDT 2015  Mon Apr 20
11:32:12 PDT 2015  COMPLETED
PayrollJob  lab1  2          2          COMPLETED  Mon Apr 20 11:32:10 PDT 2015  Mon Apr 20
11:32:11 PDT 2015  COMPLETED
PayrollJob  lab1  1          1          COMPLETED  Mon Apr 20 11:25:26 PDT 2015  Mon Apr 20
11:25:26 PDT 2015  COMPLETED

```

Get Details of a Job Execution Using `getJobExecutions`

The `getJobExecutions` attribute returns details about a particular job execution. Each entry in the collection contains an array of the elements in [Table 20-5](#).

Table 20-5 Elements in `getJobExecutions()` Attribute

Element Name	Description
JOB_NAME	The name of the batch job (String).
INSTANCE_ID	The instance ID (long).
EXECUTION_ID	The execution ID (long).
BATCH_STATUS	The batch status of this job (String).
START_TIME	The start time of the job (java.util.Date).
END_TIME	The completion time of the job (java.util.Date).
EXIT_STATUS	The exit status of the job (String).

The following example uses the `getJobExecutions()` command in a WLST script to get details for a given execution ID: `getJobExecutions(6)`. To get a list of all execution IDs, use the `getJobDetails()` method:

```

connect('admin', 'admin123')
partitionRuntime()
cd('BatchJobRepositoryRuntime')
cd('myserver')
executions=cmo.getJobExecutions(6)
print "JobName InstanceID ExecutionID BatchStatus StartTime EndTime ExitStatus"

```

```
for e in executions
  print e[0], " ", e[1], " ", e[2], " ", e[3], " ", e[4], " ", e[5], " ", e[6]
```

This is sample output after running `getJobExecutions()`:

```
JobName      InstanceID  ExecutionID  BatchStatus      StartTime
EndTime
ExitStatus
PayrollJob   6           6           COMPLETED      Fri Apr 24 10:11:00 PDT 2015  Fri Apr 24 10:11:01
PDT 2015    COMPLETED
```

Get Details of a Job Step Execution Using `getStepExecutions`

The `getStepExecutions` attribute returns metrics about each step in a job execution. Each entry in the collection contains an array of the elements in [Table 20-6](#).

Table 20-6 Elements in `getStepExecutions()` Attribute

Element Name	Description
STEP_NAME	The name of the batch job step (String).
STEP_ID	The step ID (long).
EXECUTION_ID	The execution ID (long).
BATCH_STATUS	The batch status of this job (String).
START_TIME	The start time of the job (java.util.Date).
END_TIME	The completion time of the job (java.util.Date).
EXIT_STATUS	The exit status of the job (String).

The following example shows using `getStepExecutions()` in a WLST script to get details for a given step execution ID: `getStepExecutions(6)`. To get a list of all execution IDs, use the `getJobDetails()` method.

```
connect('admin', 'admin123')
partitionRuntime()
cd('BatchJobRepositoryRuntime')
cd('myserver')
executions=cmo.getStepExecutions(6)
print "StepName      StepExecutionID  BatchStatus      StartTime      EndTime      ExitStatus"
  print e[0], " ", e[1], " ", e[2], " ", e[3], " ", e[4], " ", e[5], "
```

This is sample output after running `getStepExecutions()`:

```
StepName      StepExecutionID  BatchStatus      StartTime
EndTime
ExitStatus
PayrollJob   6           6           COMPLETED      Fri Apr 24 10:11:00 PDT 2015  Fri Apr 24 10:11:01
PDT 2015    COMPLETED
```

Configuring Partition Batch Job Runtime: Related Tasks and Links

- Configuring the Batch Job Runtime in *Administering Server Environments for Oracle WebLogic Server*.

- For information about configuring a Managed Executor Service Template, see Global CMO Templates in *Administering Server Environments for Oracle WebLogic Server*.
- For more information about configuring a JDBC data source, see Creating a JDBC Data Source in *Administering JDBC Data Sources for Oracle WebLogic Server*.
- For more information about batch-related configuration and runtime MBeans, see the [BatchConfigMBean](#), [BatchJobRepositoryRuntimeMBean](#), and the [PartitionMBean](#) in the *MBean Reference for Oracle WebLogic Server*.
- For more information about using WLST commands, see WebLogic Server WLST Online and Offline Command Reference in the *WLST Command Reference for WebLogic Server*.
- Sample Applications and Code Examples in *Understanding Oracle WebLogic Server*.
- [Batch Processing](#) in *The Java EE 7 Tutorial*.
- Java Specification Request 352: Batch Applications for the Java Platform (<http://jcp.org/en/jsr/detail?id=352>).

21

Configuring Resource Adapters

Oracle WebLogic Server Multitenant allows standalone and embedded resource adapters to be deployed in a domain partition through resource groups and resource group templates.

Note:

WebLogic Server Multitenant domain partitions, resource groups, resource group templates, virtual targets, and Resource Consumption Management are deprecated in WebLogic Server 12.2.1.4.0 and will be removed in the next release.

This chapter includes the following sections:

- [Configuring Resource Adapters in a Domain Partition: Overview](#)
- [Best Practices and Considerations When Using Resource Adapters in a Domain Partition](#)

Configuring Resource Adapters in a Domain Partition: Overview

When a resource adapter is deployed in a partition, all of its resources, such as the resource adapter beans, connection pools, and administered objects, are registered in the JNDI namespace of the partition, and their JNDI names and classloaders are isolated from other partitions' applications. Therefore, only those applications in the same partition can access the services provided by the resource adapter and only those applications in the same partition can access the classes of the resource adapter.

When a resource adapter is deployed in a domain, all of its resource adapter beans, connection pools, and administered objects are registered in the domain's JNDI namespace. These resources are visible only to applications deployed in the domain.

Example config.xml for Domain-Level Resource Adapter

You can configure a resource adapter in a resource group template and then reference it by the resource group of a partition.

In this example, a standalone resource adapter is defined in a resource group template. Each of the two partitions, `HR` and `Finance`, defines a resource group that inherits the applications and resources from the resource group template. In this case, each partition has its own resource adapter.

```
<domain>  
...
```

```

<resource-group-template>
  <name>RGT1</name>
  <app-deployment>
    <name>my_ra</name>
    <module-type>rar</module-type>
    <source-path>/some/directory/mail-connector.rar</source-path>
    <deployment-order>120</ deployment-order >
    ...
  </app-deployment>
  ...
</resource-template>

<partition>
  <name>HR</name>
  <resource-group>
    <name>RG1</name>
    <resource-group-template>RGT1</resource-group-template>
  </resource-group>
  ...
</partition>
<partition>
  <name>Finance</name>
  <resource-group>
    <name>RG2</name>
    <resource-group-template>RGT1/resource-group-template>
  </resource-group>
  ...
</partition>
...
</domain>

```

Example config.xml for Partition-Level Resource Adapter

In this example, there are two partitions, `HR` and `Finance`, defined in the domain. The `HR` partition has a resource group that defines a standalone resource adapter. In this case, the resource adapter deployed in the `HR` partition is invisible to the `Finance` partition.

```

<domain>
  ...
  <partition>
    <name>HR</name>
    <resource-group>
      <name>HR Apps and Resources</name>
      <app-deployment>
        <name>mailra</name>
        <module-type>rar</module-type>
        <source-path>/some/directory/mail-connector.rar</source-path>
        <deployment-order>120</ deployment-order >
        ...
      </app-deployment>
      ...
    </resource-group>
  </partition>

  <partition>
    <name>Finance</name>
    ...
  </partition>

```

```
...  
</domain>
```

Best Practices and Considerations When Using Resource Adapters in a Domain Partition

Oracle recommends that system administrators follow certain best practices and considerations when developing resource adapters for a multitenant environment.

Defining the Classloading Behavior for Partition-Level Resource Adapters

A partition-level standalone resource adapter is loaded by the partition classloader, which is a sub-classloader of the domain `/lib` classloader, and the parent of all application classloaders for a partition.

The `<enable-global-access-to-classes>` element in the `weblogic-ra.xml` deployment descriptor determines the classloading behavior of resource adapter classes in a partition.

See `weblogic-ra.xml` Schema in *Developing Resource Adapters for Oracle WebLogic Server*.

- If `<enable-global-access-to-classes>` is set to `false`, then the classloading behavior of the partition resource adapter classes is the same as the domain-level resource adapter classes.
- If `<enable-global-access-to-classes>` is set to `true`:
 - When the resource adapter is deployed to a domain, the resource adapter classes are loaded by the domain classloader. Therefore, the resource adapter classes are isolated from applications in other partitions.
 - When the resource adapter is deployed to a partition, the resource adapter classes are loaded by the partition classloader. Therefore, the resource adapter classes are isolated from applications in a domain partition and other partitions.

Overriding Application Configuration for a Partition

Each partition can specify a deployment plan that can be used to override the configuration defined in the resource group template. The resource adapter application should apply this deployment plan during deployment.

See [Overriding Application Configuration](#).

A deployment plan file can be specified at the application level using the `plan-path` attribute of the `AppDeploymentConfigMBean` to override the resource adapter's configuration. When a resource group of a partition references a resource adapter of a resource group template, it can override the resource group configuration through macro substitution for the specified deployment plan at the application level. The following example shows a resource group template with a macro for the deployment plan file name. There are two partitions, `HR` and `Finance`.

```

<domain>
  <resource-group-template>
    <name>RGT1</name>
    <app-deployment>
      <name>mailra</name>
      <name>my_ra</name>
      <module-type>rar</module-type>
      <source-path>/some/directory/sample_connector.rar</source-path>
      <deployment-order>120</ deployment-order >
      <plan-path>${MYRA-PLAN-FILE}</plan-path>
    </app-deployment>
    ...
  </resource-group-template>

  <partition>
    <name>HR</name>
    <resource-group>
      <name>RG1</name>
      <resource-group-template>RGT1</resource-group-template>
    </resource-group>
    <partition-properties>
      <partition-property>
        <name>MYRA-PLAN-FILE</name>
        <value>/apps/plans/hr/my_ra-plan.xml</value>
      </partition-property>
    </partition-properties>
    ...
  </partition>

  <partition>
    <name>Finance</name>
    <resource-group>
      <name>RG2</name>
      <resource-group-template>RGT1</resource-group-template>
    </resource-group>
    <partition-properties>
      <partition-property>
        <name>MYRA-PLAN-FILE</name>
        <value>/apps/plans/Finance/my_ra-plan.xml</value>
      </partition-property>
    </partition-properties>
    ...
  </partition>
</domain>

```

Considerations for Resource Adapter Resource Definitions

You can use the `@ConnectorResourceDefinition` and `@AdministeredObjectDefinition` annotations for defining resource adapter resources in your web module, EJB module, or equivalent application deployment descriptor. When you use these annotations, the following conditions apply in a multitenant environment:

- If `@ConnectorResourceDefinition` and `@AdministeredObjectDefinition` or equivalent deployment descriptors are defined in the domain-level application, then its related resource adapter should be deployed at the domain level.
- If `@ConnectorResourceDefinition` and `@AdministeredObjectDefinition` or equivalent deployment descriptors are defined in the partition-level application, then its related resource adapter should be deployed in the same partition.

Resource Group Migration and Resource Adapters

A standalone resource adapter is shared across resource groups in the same partition. It can be defined in a resource group, but it is not occupied exclusively by that resource group. Suppose resource group RG1 depends on a standalone resource adapter RA2 that is defined in resource group RG2; if RG2 is migrated to new cluster, then RG1 has to be migrated to that cluster at the same time. Otherwise RG1 will not work.

Oracle recommends that you define a resource adapter and all related applications in the same resource group if you want to do resource group migration.

Exporting and Importing Partitions

Exporting and importing partitions lets you easily move partitions from one domain to another, including the applications that are deployed to the partition. This feature is useful for replicating partitions across domains and for moving domains from a development to a production environment.

 **Note:**

WebLogic Server Multitenant domain partitions, resource groups, resource group templates, virtual targets, and Resource Consumption Management are deprecated in WebLogic Server 12.2.1.4.0 and will be removed in the next release.

This chapter includes the following sections:

- [Exporting and Importing Domain Partitions: Overview](#)
- [Exporting and Importing Domain Partitions: Main Steps and WLST Examples](#)
- [Exporting and Importing Domain Partitions: Related Tasks and Links](#)

Exporting and Importing Domain Partitions: Overview

Using WLST (online or offline), a Representational State Transfer (REST) API (online only), or the WebLogic Server Administration Console (online only), system administrators can export a domain partition and then import a previously exported partition into a different domain.

The following sections provide detailed information about the exporting and importing operations.

About Exporting Partitions

When a partition is exported from the source domain, it is packaged in a partition archive file that includes:

- The partition configuration
- Any resource groups contained in the partition
- Any resource group templates referred to by those resource groups
- The contents of the partition's file system, `<partition-file-system>/config` directory
- Optionally, application binary files and configurations for applications deployed to the partition

No application runtime state, nor application-specific runtime configuration is included in the partition archive file. Other examples of what would not be exported are JMS messages in queues and users in an embedded LDAP realm.

Creating the Partition Archive File

You can use the WLST `exportPartition` command to create a partition archive file, `<PartitionName>.zip`. It copies a partition's configuration into the archive file as well as (optionally) the partition's applications and libraries. This command also creates the `attributes.json` file that you can use to modify the partition's configuration on import.

The command syntax is:

```
exportPartition(partitionName, expArchPath, [includeAppsNLibs], [keyFile])
```

partitionName is the name of the partition to export, *expArchPath* is the full path to the directory in which to save the partition archive file, *includeAppsNLibs* (optionally), specifies whether to include application and library binary files in the partition archive file, and *keyFile* (optionally), provides the full path to a file containing a string to use as the encryption key to encrypt attributes in the partition archive file. See [Exporting Domain Partitions: WLST Example](#).

Partition Archive File Contents

A partition archive is a `.zip` file containing these specific partition-related files:

Files and Directories	Description
<code>partition-config.xml</code>	Contains the partition configuration and resource group templates configuration from <code>config.xml</code> .
<code>MANIFEST.MF</code>	Includes a time stamp and domain information for the archive file.
<code><PartitionName>-attributes.json</code>	Contains the MBean attributes and properties which can be changed by the administrator while importing the partition. In addition to being contained in the archive file, a copy of this file is also located with the archive file in the file system to make it easier to update.
<code>expPartSecret</code>	Contains the encrypted secret key used for encrypting and decrypting encrypted attributes.
<code>domain/config/resource-group-templates/<resource_template_name>/*</code>	Contains files related to any resource group template associated with this partition.
<code>domain/config/partitions/<partition-name>/*</code>	Specifies the configuration files under <code>domain/config/partitions/<partition-name>/config/*</code> .
<code>pfs/config/*</code>	Contains the <code><partition-file-system>/config</code> directory content. This would also include system resources (JMS, JDBC, and such) descriptor files.

Files and Directories	Description
domain/upload/resource-group-templates/<resource_template_name>/<application_name>/	Contains resource group template-level binary files.
pfs/upload/<application_name>/	Contains resource group-level application binary files.

About Importing Partitions

A prerequisite to importing a partition is that the server instance must already have a domain configured. When a partition is imported into a new domain, some external systems may need to be configured to be aware of the newly imported partition. While importing a partition archive file, the system administrator may need to update the dependencies on the domain (like virtual targets, security realms, and resource group templates) and also optionally update other attributes in the partition configuration to make it valid, such as partition properties, JDBC, JMS, and other resources in resource groups. In addition, applications and system resources need to be deployed for the new partition.

During the import operation, the system administrator can override specific portions of the partition configuration by supplying a <PartitionName>-attributes.json file with modified attributes suitable for the target domain. However, changing the value of the name attribute is not supported; for example, "name" : "P1" in the following sample <PartitionName>-attributes.json cannot be changed.

```
{
  "partition" : {
    "name" : "P1",
    "jdbc-system-resource-override" : {
      "URL" : "url.com",
      "Id" : "0",
      "name" : "test123",
      "CachingDisabled" : "false",
      "Registered" : "false",
      "User" : "test123",
      "DynamicallyCreated" : "false",
      "DataSourceName" : "test-source"
    },
    "resource-group-template" : {
      "name" : "RGT1",
      "jdbc-system-resource" : {
        "name" : "jdbc1",
        "descriptor-file-name" : "jdbc/P1DB1-8882-jdbc.xml"
      }
    },
    "jms-server-resource" : {
      "name" : "jms1",
      "PagingDirectory" : ""
    }
  }
}
```


In the prior example, all the `JDBCSystemResourceOverrideMBean` attributes are copied to the `<PartitionName>-attributes.json` file along with their current values.

If a resource group template already exists in the target domain, then you must use the `createNew` option to overwrite the existing resource group template and create a new one using a new name. (See Step 2 in [Importing Domain Partitions: Main Steps](#).)

Customizing Exporting and Importing System Resources

When domain partitions are exported, partition and resource group template configuration attributes are present in the `attributes.json` file. System resources referenced by the partition being exported also appear in `attributes.json`.

WebLogic Server MT lets you export all the system resource attributes of a partition into a JSON file so you can modify those attributes and then import them with the partition. Being able to override system resources is particularly useful in PaaS use cases where a partition has resource groups and system resources which are deployed directly to the resource group (as compared to SaaS use cases where resource groups are typically derived from a resource group template).



Note:

Oracle advises that you use caution when overriding partition attributes. If you make a change that is not consistent with the entire configuration, you might encounter start up or later configuration validation issues.

The following `attributes.json` file shows an example of JDBC system resource elements.

```
"jdbc-system-resource": {
  "name": "mydb",
  "jdbc-data-source": {
    "@xmlns": "http://xmlns.oracle.com/weblogic/jdbc-data-
source",
    "@xmlns:sec": "http://xmlns.oracle.com/weblogic/security",
    "@xmlns:wls": "http://xmlns.oracle.com/weblogic/security/
wls",
    "@xmlns:xsi": "http://www.w3.org/2001/XMLSchema-instance",
    "@xsi:schemaLocation": "http://xmlns.oracle.com/weblogic/
jdbc-data-source http://xmlns.oracle.com/weblogic/jdbc-data-source/
1.0/jdbc-data-source.xsd",
    "name": "mydb",
    "jdbc-driver-params": {
      "url": "jdbc:derby:memory:mydb",
      "properties": {"property": [
        {
          "name": "user",
          "value": "scott"
        },
        {
          "name": "databaseName",
          "value": "prodMydb"
        }
      ]}
    }
  }
}
```

```

    }
  },
  "password-encrypted": "{AES}LbwIP8Ca\
RYAv4z599NAqM0eyHCcCl2um3gR6tziNpM="
}
}
}

```

For example, to make the resources suitable for a production domain, you might edit the user value from `scott` to `ScottProd` (a user for a production domain) and the URL from `jdbc:derby:memory:mydb` to `jdbc:derby:memory:dbprod`. When you import the partition, the system resources reflect the new values contained in the `attributes.json` file.

Passwords are exported encrypted. After importing, the password is placed in the `config.xml` file, encrypted with the domain specific key from the importing domain. If you want to change the password, you must:

1. Create a new encrypted password using the `weblogic.security.Encrypt` utility from the domain where the partition is being imported.
2. Replace the password in the JSON file with the one you created in step 1.
3. Import the partition.

After importing, the password that you put in the JSON file will be used for the system resource.

Exporting and Importing Applications

When a partition is exported, whether or not application binary files are included in the partition archive file is determined by the value of the `includeApps` option on the export operation. If `includeApps` is `true`, then the binary files are included. If `false`, then they are excluded. The `<PartitionName>-attributes.json` file contains attributes for the application or library source path, deployment plan path, and staging mode. You can change these attributes in the `<PartitionName>-attributes.json` file.

The `includeApps` option also affects how the application is deployed upon being imported:

- If `true` (default), then the deployment files are copied to `<domain-dir>/<server>/upload` or `<partition-dir>/<server>/upload`, depending on whether it is a resource group template-level application or resource group-level application, and deployed using the original staging mode. If the staging mode is `NOSTAGE` or `EXTERNAL` (on the targeted server), then the system administrator is responsible for making the application available from the correct location in the target domain.
- If `false`, then the system administrator is responsible for making the application available from the correct location in the target domain.

Exporting and Importing Encrypted Attributes

During the export operation, a new secret key is generated and stored in the `expPartSecret` file in the exported `<PartitionName>.zip` file. All the encrypted attributes in the `partition-config.xml` file and any system resource descriptors that are part of `<PartitionName>.zip` will be encrypted using the new secret key in the

`expPartSecret` file. Alternatively, you can provide your own secret key by using the `keyFile` option.

During the import operation, after the entire partition is read and the attributes changed in accordance with the `<PartitionName>-attributes.json` file, the following steps are used to process encrypted attributes:

1. The secret key in the `expPartSecret` file in the exported `.zip` is read.
2. The secret key is decrypted using a second key that is either the default key in the WebLogic Server source, or the key provided using the `keyFile` option to import. The user-provided key must match the key used on import.
3. All the encrypted attributes in the partition MBean (read and modified in the export operation) are decrypted using the key read from the `expPartSecret` file.
4. All the encrypted attributes in the partition MBean are encrypted with the domain-specific key (`SerializedSystemIni.dat`) for the imported domain.

Exporting and Importing Domain Partitions: Main Steps and WLST Examples

You can export a domain partition (the source domain) with its entire configuration and data, as a partition archive file. With few configuration changes, you can then import the partition archive file into another instance of WebLogic Server MT (the target domain).

You might need to update domain dependencies, such as virtual targets and security realms, and optionally update other attributes in the partition configuration to make it valid.

Exporting Domain Partitions: Main Steps

Prior to exporting a domain partition, you must first have created or imported one.

The main steps for exporting domain partitions are as follows:

1. Select the domain partition that you want to export.
2. Provide the full path to the directory in which to save the partition archive file.

Each domain partition should be located in its own directory. The domain partition archive file is overwritten if it already exists in the specified location. Other files in the directory may be overwritten as well.
3. Optionally, select to include the installed application and library binary files in the exported partition archive file.
4. Optionally, enter the full path to a file containing a string to use as the encryption key to encrypt attributes in the partition archive file.

If you do not provide your own key, then a new secret key will be generated and stored in the `expPartSecret` file in the exported `<PartitionName>.zip` file.

Exporting Domain Partitions: WLST Example

The following WLST command exports `partition-1` to the `/var/tmp` directory. Application and library binary files are not included. `/home/foo/mykeyfile` is the path to the `mykeyfile` encryption key.

```
wls:/mydomain/serverConfig> exportPartition("partition-1", "/var/tmp/", false, "/home/foo/mykeyfile")
```

Importing Domain Partitions: Main Steps

Prior to importing a domain partition:

- You must have previously exported a domain partition (the source domain) to a partition archive file.
- The server instance must already have a domain configured (the target domain).

The main steps for importing domain partitions are as follows:

1. Provide the full path to the partition archive file that you want to import.
2. Optionally, select the **overwrite existing resource group templates** option if the resource group template already exists in the target domain and you want to create a new one using a new name.

All resource group templates used by the source domain partition are contained in the partition archive file and are imported along with the partition into the target domain. If the resource group template already exists in the target domain and you do not specify to overwrite the existing resource group template, then the import operation will fail.

3. Optionally, specify a name to use for the partition when it is created in the target domain. This defaults to the original name of the partition.
4. Optionally, enter the full path to a file containing the key to decrypt attributes in the partition archive file.

Importing Domain Partitions: WLST Example

The following WLST command imports the `partition-1` archive file located in the `/var/tmp` directory. Application and library binary files are not included. The file `/home/foo/mykeyfile` is used as the encryption key.

```
wls:/mydomain/serverConfig> importPartition("/var/tmp/partition-1.zip", keyFile="/home/foo/mykeyfile")
```

Exporting and Importing Domain Partitions: Related Tasks and Links

- Export domain partitions and Import domain partitions in *Administering Oracle WebLogic Server with Fusion Middleware Control*
- [Export partitions](#) and [Import partitions](#) in *Oracle WebLogic Server Administration Console Online Help*

Managing Named Concurrent Edit Sessions

In previous releases, WebLogic Server configuration edit sessions supported only one active edit session at a time. The system administrator got a global edit lock, made changes, and then activated them. Other administrators could not make changes at the same time. In this release, there are situations in which more than one administrator may need to make configuration changes. A WebLogic Server MT domain contains multiple partitions, each with its own administrator. Partition administrators must be able to make configuration changes to the partition configuration and resources deployed in the partition without affecting other partition administrators or the WebLogic Server system administrator. Therefore, WebLogic Server has enabled multiple, concurrent edit sessions that support one or more configuration edit sessions per partition plus global configuration edit sessions.

 **Note:**

WebLogic Server Multitenant domain partitions, resource groups, resource group templates, virtual targets, and Resource Consumption Management are deprecated in WebLogic Server 12.2.1.4.0 and will be removed in the next release.

This chapter includes the following sections:

- [Named Concurrent Edit Sessions: Overview](#)
- [Managing Named Concurrent Edit Sessions: Main Steps](#)
- [Managing Named Concurrent Edit Sessions: WLST Example](#)
- [Managing Named Concurrent Edit Sessions: Related Tasks and Links](#)

Named Concurrent Edit Sessions: Overview

With this feature, an administrator creates a named edit session, makes changes, and then activates the changes. Another administrator also creates a named edit session in parallel. If there are conflicts between the edit session from the first administrator and the changes made by a second administrator, then the second administrator receives an error when activating the changes. The second administrator can then resolve the conflicts and activate the desired changes.

For details, see [Managing Named Concurrent Edit Sessions: Main Steps](#).

Managing Named Concurrent Edit Sessions: Main Steps

Follow these main steps for managing named concurrent edit sessions.

1. Administrators create (and destroy) named edit sessions.

You use the `ConfigurationManagerMBean` API to start an edit session, activate changes, show changes, undo changes, and so forth.

Each edit session has its own configuration files and `ConfigurationManagerMBean` instance.

2. Changed configuration files are retained in an edit-session-specific directory:
 - Global named edit session changes are retained in a subdirectory of a new edit directory at the domain level: `<domain-directory>/edit/<edit-session-name>`, where `<edit-session-name>` is a version of the edit session name specified when the edit session was created.
 - Partition-specific named edit session changes are retained in a subdirectory of the partition directory: `<partition-directory>/edit/<edit-session-name>`, where `<edit-session-name>` is a version of the edit session name specified when the edit session was created.
3. Conflicts may occur between parallel edit sessions when changes are activated. If any conflicts occur during activation, then the activate operation fails and you must resolve the conflicts manually.

Use the `resolve` method to apply any concurrently applied configuration changes to the current edit session.

After calling the `resolve` method, you can then proceed with the activation of pending changes in the edit session.

To resolve conflicts, administrators can use these informational messages:

- Each conflict contains a description including the identification of MBeans or properties in conflict.
- Each conflict also contains a message describing the resolution for this conflict.
- Resolution log messages contain short descriptions of each resolution and merge step. Administrators can use the resolution log message to identify what was modified and how it was modified in the current edit session.

Managing Named Concurrent Edit Sessions: WLST Example

You can use WLST to manage configuration edit sessions.

See the following WLST script example.

```
connect("username", "password")

# Enter existing named edit session or create one and enter it.
edit("foo")
startEdit()
cmo.createServer("Server-1")

# Create a different edit session.
edit("bar")
startEdit()
ls("Servers") #Server-1 is not printed because it is created in an independent edit
session foo
```

```

sl=cmo.createServer("Server-1")
sl.setListenPort(5555) # Conflicting modification - the same server with a different
port

# Go back to edit session foo and activate.
edit("foo")
activate()

# List edit sessions / list with details.
showEditSession() # Lists all edit sessions
showEditSession("bar") # More detailed information about edit session bar

# Go back to edit session bar and activate.
edit("bar")
activate() # Conflict occurs

# Resolve conflicts based on a standard resolution strategy.
resolve()
activate()

# Edit sessions can also be destroyed.
destroyEditSession("foo")
destroyEditSession("bar")

```

The following example shows the edit session conflict resolution in detail:

```

wls:/wls/edit(foo)/> showEditSession() # Lists all edit sessions
List of named edit sessions [for details use showEditSession(<name>)]:
  default
  bar
  foo
wls:/wls/edit(foo)/> showEditSession("bar") # More detailed information about edit
session bar
bar
  Creator: wls
  Editor (lock owner): wls
  Resolve recommended: Yes
  Contains unactivated changes: Yes

wls:/wls/edit(foo)/> # Go back to edit session bar and activate.
wls:/wls/edit(foo)/> edit("bar")
You already have an edit session in progress and hence WLST will
continue with your edit session.
Other configuration changes were activated. Call resolve() to merge it into this
edit tree.
wls:/wls/edit(bar)/ !> activate() # Conflict occurs.
Activating all your changes, this may take a while ...
The edit lock associated with this edit session is released after the activation is
completed.
Traceback (innermost last):
  File "<console>", line 1, in ?
  File "<iostream>", line 471, in activate
  File "<iostream>", line 553, in raiseWLSTException
WLSTException: Error occurred while performing activate : Error while Activating
changes. : 1 conflict:
[1]
[wls]/Servers[Server-1] - a bean with the same qualified name has already been
added to [wls].
Description of resolution operation:
  A bean added by this session overrides the one present in the current configuration.

```

```
wls:/wls/edit(bar)/> # Resolve conflicts based on a standard resolution strategy.
wls:/wls/edit(bar)/> resolve()
1 conflict:
[1]
[wls]/Servers[Server-1] - a bean with the same qualified name has already been
added to [wls].
Description of the resolution:
  A bean added by this session overrides the one present in the current configuration.
Patch:
No difference
wls:/wls/edit(bar)/ !> activate()
Activating all your changes, this may take a while ...
The edit lock associated with this edit session is released after the activation is
completed.
Activation completed

wls:/wls/edit(bar)/> # Edit sessions can also be deleted.
wls:/wls/edit(bar)/> destroyEditSession("foo")

wls:/wls/edit(bar)/> destroyEditSession("bar")
Current edit tree is being removed; redirecting WLST cursor location to the config
runtime tree.
```

Managing Named Concurrent Edit Sessions: Related Tasks and Links

- Manage edit sessions and View and resolve conflicts in *Administering Oracle WebLogic Server with Fusion Middleware Control*
- [View pending changes](#) and [Resolve conflicts](#) in *Oracle WebLogic Server Administration Console Online Help*

Configuring Transactions

WebLogic Server MT supports transaction processing in a multiple partition environment. The scope of the transactions can be either at the domain or partition level.

 **Note:**

WebLogic Server Multitenant domain partitions, resource groups, resource group templates, virtual targets, and Resource Consumption Management are deprecated in WebLogic Server 12.2.1.4.0 and will be removed in the next release.

This chapter includes the following sections:

- [Configuring Transactions in a Domain Partition: Overview](#)
- [Configuring LLR Data Source, JDBC TLog Data Source, and a Determiner Resource in a Domain Partition: Limitations](#)
- [Stateful EJB Resource Group Migration Failover: Limitations](#)

Configuring Transactions in a Domain Partition: Overview

Most of the attributes that define a transaction environment are defined at the global (domain) level. However, the Timeout Seconds option for JTA needs to be configured at the partition level. See Domain Partition: Configuration: JTA in the [Oracle WebLogic Server Administration Console Online Help](#).

Configuring LLR Data Source, JDBC TLog Data Source, and a Determiner Resource in a Domain Partition: Limitations

A Logging Last Resource (LLR) data source, JDBC TLog data source, or a Determiner Resource cannot be registered in a partition. In this release of WebLogic Server, any attempt to register these resources at the partition level will result in a system exception.

This restriction is because a WebLogic Server instance cannot start if any of these resources is not available. If a server is allowed to start despite the inaccessibility of a resource (and hence its commit records), the recovery of resources enlisted in transactions that also enlisted those resources may be non-atomic as the lack of a commit record would result in a rollback of those resources.

If you still want to use LLR for the resource commit ordering it inherently provides, then use the *first resource commit* functionality provided in the WebLogic Server 12.2.1

release. Because this feature requires a true XA resource rather than the local-transaction-based LLR resource, it does not provide the performance benefits that LLR offers and it does not require an LLR table. LLR data sources are pinned to the server that they are deployed on, but XA data sources are not. Therefore, in the case where multiple servers are enlisted in a transaction, the two-phase commit flow may be slightly different. See First Resource Commit Ordering in *Developing JTA Applications for Oracle WebLogic Server*.

Stateful EJB Resource Group Migration Failover: Limitations

Stateful EJB client failover after a live migration of its resource groups to another cluster takes longer than a failover within a cluster because the cluster service waits for the result of the replica search from the cluster members with `ClusterMBean.getSessionStateQueryRequestTimeout()`, which has a default value of 30 seconds. Therefore, during the failover of a stateful EJB invocation after a live migration of its resource group to another cluster, if there is a transaction on the thread, the transaction may time out during the failover of the stateful EJB method invocation if the transaction timeout value is not sufficiently long enough. The default transaction timeout is 30 seconds.

To avoid transaction timeout during a resource group live migration across clusters that involve a stateful EJB client, increase the transaction timeout value before the resource group migration. You can then reset the timeout to its previous value when all the clients that use the stateful EJB that are migrated have failed over to the target cluster. For example, if there is an EJB that invokes the stateful EJB in a container managed transaction, increase the `trans-timeout-seconds` deployment descriptor for the EJB application using the WebLogic Server Administration Console. For more information see, [EJB: Configuration](#).

Configuring Web Services

Learn how to configure the resources required to support advanced web services and SOAP over JMS in Oracle WebLogic Server Multitenant (MT) using WebLogic Scripting Tool (WLST) scripts. Also, the specific configurations that are required when using web services in a WebLogic Server MT environment.

 **Note:**

WebLogic Server Multitenant domain partitions, resource groups, resource group templates, virtual targets, and Resource Consumption Management are deprecated in WebLogic Server 12.2.1.4.0 and will be removed in the next release.

This chapter includes the following sections:

- [Configuring Web Services: Overview](#)
- [Configuring Web Services: Main Steps](#)
- [Configuring Web Services: Using Partitioned Distributed Topics with SOAP over JMS](#)
- [Configuring Web Services: Related Tasks and Links](#)

Configuring Web Services: Overview

WebLogic Server includes two WLST scripts that provide the ability to configure the resources required for certain JAX-WS web service features in a WebLogic Server MT environment.

- `wlsws-advanced-jaxws-mt-config.py`: Configures advanced JAX-WS web services including asynchronous messaging, web services reliable messaging, message buffering, web services atomic transactions, and security using WS-SecureConversation.
- `wlsws-soapjms-mt-config.py`: Configures SOAP over JMS transport.

Both scripts are located in the `oracle_home/oracle_common/webservices/bin/` directory, where `oracle_home` is the directory that you specified as Oracle home when you installed WebLogic Server.

Configuring Web Services: Main Steps

Follow these main steps to configure web service resources in a WebLogic Server MT environment.

1. If you have not already done so, create a domain partition. See [Configuring Domain Partitions](#).

2. If you have not already done so, create a resource group in the partition. See [Configuring Resource Groups](#).
3. At the domain level, create a user configuration file and an associated key file using the `storeUserConfig` WLST command. See `storeUserConfig` in *WLST Command Reference for WebLogic Server*.
4. Execute one or both of the WLST scripts from the `JAVA_HOME/bin/java` directory. Note that you must include the location of the `weblogic.jar` file in the class path.

For example, to configure the resources required for the WebLogic Server MT advanced web services for JAX-WS, use the following command:

```
JAVA_HOME/bin/java -classpath weblogic.jar_location weblogic.WLST ./wlsws-advanced-jaxws-mt-config.py
-myUserConfigFile userConfigFile -myUserKeyFile userKeyFile
-myURL AdminServer_t3_url -partitionName partitionName
-rgName resourceGroupName -isCluster true_or_false
-middlewreHome middlewareHomeDir
```

To configure the resources required in WebLogic Server MT for SOAP over JMS transport, use the following command:

```
JAVA_HOME/bin/java -classpath weblogic.jar_location weblogic.WLST ./wlsws-soapjms-mt-config.py
-myUserConfigFile userConfigFile -myUserKeyFile userKeyFile
-myURL AdminServer_t3_url -partitionName partitionName
-rgName resourceGroupName -isCluster true_or_false
-middlewreHome middlewareHomeDir
```

Both scripts require the command options described in [Table 25-1](#).

Table 25-1 Required Options for Advanced Web Services WLST Scripts

Option	Description
<code>-myUserConfigFile</code>	Name of the file that you specified for the <code>userConfigFile</code> argument when you executed the <code>storeUserConfig</code> WLST command in Step 3. This file stores the user configuration.
<code>-myUserKeyFile</code>	Name of the file that you specified for the <code>userKeyFile</code> argument when you ran the <code>storeUserConfig</code> WLST command in Step 3. This file stores the key information that is associated with the specified user configuration file.
<code>-myURL</code>	T3 protocol URL for the Administration Server for the partition, for example <code>t3://host:port</code> .
<code>-partitionName</code>	Name of the partition in which the resources are being configured.
<code>-rgName</code>	Name of the resource group in the partition.
<code>-isCluster</code>	Argument that indicates if the partition is in a cluster. Valid values are: <ul style="list-style-type: none"> • <code>true</code>: Partition is in a cluster. • <code>false</code>: Partition is not in a cluster.

Table 25-1 (Cont.) Required Options for Advanced Web Services WLST Scripts

Option	Description
-middlewareHome	Location of the directory that you specified as the Oracle home directory when you installed WebLogic Server.

Configuring Web Services: Using Partitioned Distributed Topics with SOAP over JMS

SOAP over JMS supports both queues and topics as the JMS destination type. The type that you use is determined by the application. When using a uniform distributed topic as the request destination for SOAP over JMS in a WebLogic Server MT environment, you must use a partitioned uniform distributed topic.

In WebLogic JMS, you can configure distributed topics as replicated distributed topics or partitioned distributed topics. While replicated distributed topics work with `topicMessageDistributionMode` settings of `Compatibility`, `One-Copy-Per-Server`, or `One-Copy-Per-Application`, partitioned distributed topics work only with `One-Copy-Per-Server` or `One-Copy-Per-Application`.

To configure a partitioned uniform distributed topic for SOAP over JMS in WebLogic Server MT, configure the `@JMSTransportService` annotation, and set the `topicMessageDistributionMode` configuration property on the `activationConfig` property to `One-Copy-Per-Server` or `One-Copy-Per-Application`. The `Compatibility` value is not supported for partitioned distribution topics.

For example:

```
@JMSTransportService(targetService="poNotifyService"
,
destinationName="com.oracle.webservices.jms.SoaJmsRequestTopic"
, destinationType=JMSDestinationType.TOPIC
, activationConfig =
"topicMessagesDistributionMode=One-Copy-Per-Application"
, jndiURL = "t3://@wls-server@")
```

See *Using SOAP Over JMS Transport* in *Developing JAX-WS Web Services for Oracle WebLogic Server*.

 **Note:**

You must configure the JMS topic before deploying the SOAP over JMS application using the partitioned distributed topic. See *Configuring Partitioned Distributed Topics* in *Administering JMS Resources for Oracle WebLogic Server*.

Configuring Web Services: Related Tasks and Links

- [Configuring Domain Partitions](#)

- [Configuring Resource Groups](#)
- *Developing JAX-WS Web Services for Oracle WebLogic Server*
- *storeUserConfig in WLST Command Reference for WebLogic Server*
- *Creating and Configuring Servlets in Developing Web Applications, Servlets, and JSPs for Oracle WebLogic Server*

Monitoring and Debugging Partitions

Learn how to monitor and debug partitions in Oracle WebLogic Server Multitenant (MT).

 **Note:**

WebLogic Server Multitenant domain partitions, resource groups, resource group templates, virtual targets, and Resource Consumption Management are deprecated in WebLogic Server 12.2.1.4.0 and will be removed in the next release.

This chapter includes the following sections:

- [Monitoring Domain Partitions: Overview](#)
- [Partition Log and Diagnostics Data](#)
- [Configuring Partition-Scoped Logging](#)
- [Configuring Partition-Scoped Debugging](#)
- [Configuring Diagnostic System Modules](#)
- [Accessing Diagnostic Data](#)
- [Monitoring Resource Consumption Management](#)
- [Instrumenting Partition-Scoped Applications](#)
- [Configuring Partition-Scoped Diagnostic Image Capture](#)
- [WLST Diagnostic Commands for Partition Administrators](#)

Monitoring Domain Partitions: Overview

The WebLogic Diagnostics Framework (WLDF) provides these partition-scoped diagnostic capabilities: logging, debugging, monitoring of partition resources, viewing log data, instrumenting partition-scoped applications, and diagnostic image capture.

- **Logging**

The logs for several WebLogic Server components, such as partition-scoped JMS, SAF, and servlet resources, are available in the partition file system directory. In a multitenant environment, partition users can configure partition-scoped loggers and control the levels of these loggers without affecting other partitions. See [Configuring Partition-Scoped Logging](#).

 **Note:**

The logs for server- and domain-scoped resources, such as the server-scoped HTTP access log, the WLDF Harvester component, the Instrumentation component, and also the server and domain logs, are available only from the WLDF Data Accessor. See [Accessing Diagnostic Data](#).

- **Debugging**
In coordination with the system administrator, a partition user can set debug flags on WebLogic Server components so that debug messages can be broadcast on behalf of work performed for that partition, without affecting other partitions. See [Configuring Partition-Scoped Debugging](#).
- **Monitoring of partition resources**
Diagnostic system modules can be configured in a resource group, or in a resource group template, to enable harvesting of partition-specific metrics and to configure policies and actions on partition-scoped resources. See [Configuring Diagnostic System Modules](#).

WLDF also provides the ability to gather partition-scoped resource consumption metrics, which are exposed as attributes on the `PartitionResourceMetricsRuntimeMBean`. See [Monitoring Resource Consumption Management](#).
- **Viewing log data**
Partition administrators do not have file-level access to log files, but they can use the WLDF Data Accessor and supported WLST functions. See [Accessing Diagnostic Data](#).
- **Instrumenting application instrumentation**
Applications deployed within a partition may be instrumented using the WLDF Instrumentation component. See [Instrumenting Partition-Scoped Applications](#).
- **Diagnostic image capture**
Image capture can be initiated either manually by a partition-scoped user, or by a policy configured in a partition-scoped diagnostic system module. Only the content specific to the partition is included in the generated diagnostic image. See [Configuring Partition-Scoped Diagnostic Image Capture](#).

Partition Log and Diagnostics Data

WebLogic Server maintains logs and diagnostics data files to track events and activity performed on behalf of partitions. Partition-specific log files are located in the `partitions/<partition>/system/servers/<server-name>` directory.

Log File	Contents
ServerLog	Log records for Weblogic Server events generated when doing work within the scope of a partition. The server log is not available within the partition file system; it is available only through the WLDF Data Accessor.

Log File	Contents
DomainLog	Log records for domain-scoped events generated on behalf of a partition. The domain log is not available within the partition file system: it is available only through the WLDF Data Accessor.
HTTPAccessLog (partition-scoped virtual target access log)	The <code>access.log</code> file for each virtual target's web server, maintained separately for each partition and can be accessed by the partition user.
HarvestedDataArchive	MBean metrics applicable to a partition collected by the WLDF Harvester component. This file is available only through the WLDF Data Accessor.
EventsDataArchive	Events generated by the WLDF Instrumentation component. This file is available only through the WLDF Data Accessor.
JMSMessageLog (partition-scoped)	Log records for partition-scoped JMS resources. An individual log file is maintained for each partition and may be accessed by the partition user.
JMSSAFMessageLog (partition-scoped)	Log records for partition-scoped SAF agent resources. An individual log file is maintained for each partition and may be accessed by the partition user.
DataSourceLog	Data source profile records generated when doing work within the scope of a partition.
WebAppLog (partition-scoped)	Servlet context logs from partition-scoped applications. An individual log file is maintained for each partition and may be accessed by the partition user.
ConnectorLog (partition-scoped)	Java EE Connector Architecture resource adapter logs from partition-scoped resource adapters. An individual log file is maintained for each partition and may be accessed by the partition user.

Configuring Partition-Scoped Logging

WebLogic Server MT allows configuration of levels for `java.util.logging` loggers that are used by applications deployed to a partition.

Within a WebLogic Server domain, different partitions may contain copies of the same application. In such cases, different instances of the application may use `java.util.logging` loggers with same name. The application may have dependencies on libraries that use `java.util.logging` loggers and that are on the system class path. In a multitenant environment, partition users can control the levels of these loggers within the scope of their partition without affecting other partitions.

The ability to configure `java.util.logging` logger levels for partition-scoped resources is enabled by the following:

- A custom log manager, `WLogManager`, which the system administrator must configure as the global log manager for the domain
- The `PartitionLogMBean.PartitionLoggerLevels` attribute, which partition users can configure by specifying key-value pairs consisting of logger names and the corresponding `java.util.logging.Level` names

To enable the ability for partition users to configure partition-scoped logger levels:

1. The system administrator must configure the `WLogManager` for the domain by including the following `weblogic.Server` option in the WebLogic Server start command:

```
-Djava.util.logging.manager=weblogic.logging.WLogManager
```

2. The partition user configures key-value pairs in the `PartitionLogMBean.PartitionLoggerLevels` attribute.

The following example shows using WLST to configure partition-scoped logging and debugging on partition `p1`.

```
startEdit()  
cd('/')  
p1 = cmo.createPartition('p1')  
plog = p1.getPartitionLog()  
plog.addEnabledServerDebugAttribute('DebugJNDI')  
props = java.util.Properties()  
props.put('foo.bar.logging', 'WARNING')  
plog.setPlatformLoggerLevels(props)  
save()  
activate()
```

For more information about debugging, see [Configuring Partition-Scoped Debugging](#).

Configuring Partition-Scoped Debugging

WebLogic Server MT supports debugging of partition-scoped resources.

- The `ServerDebugMBean` includes the `PartitionDebugLoggingEnabled` attribute. This attribute controls whether partition-scoped debugging is enabled, and may be accessed only by the system administrator.

This attribute is disabled by default. When enabled, partition-scoped debugging is available for all partitions in the domain.
- The `PartitionLogMBean` includes the `EnabledServerDebugAttributes` attribute, for which a partition user can define, as an array of `Strings`, any of the debug flags available for the `ServerDebugMBean`.
- When debugging is enabled in the domain and configured in a partition, debug messages that are broadcast by a system resource doing work on behalf of the partition are sent to that partition.

 **Note:**

Oracle strongly recommends that when troubleshooting a server problem, the partition user and the system administrator should consult one another prior to enabling server-level debugging for a partition. Furthermore, partition-scoped debugging should be enabled only for a short period of time.

A typical use case for partition-scoped debugging involves consultation with Oracle Support Services to identify the specific debug flags that the partition user should configure on the `PartitionLogMBean.EnabledServerDebugAttributes` attribute.

See the preceding example for an example WLST script that shows configuring partition-scoped debugging.

Configuring Diagnostic System Modules

You can include WLDF diagnostic system modules in resource groups and resource group templates to allow partition-specific monitoring. Note that not all functionality provided by domain-scoped diagnostic system modules is enabled when scoped to a partition. In particular, the server-level WLDF Instrumentation component, which affects the server as a whole, is not available at the partition level.

The following sections explain how to configure partition-scoped diagnostic system modules:

- [Metrics Harvesting](#)
- [Configuring Policies and Actions](#)

Metrics Harvesting

The WLDF Harvester component periodically samples JMX runtime MBean attributes consistent with its configuration in a diagnostic system module and stores the data in the Archive. This capability is available within the scope of a partition, but the diagnostic system module:

- May access the runtime MBeans for resources within the same partition only. Partition users have read-only access to certain monitoring information from MBeans at the global level. These MBean attributes can be harvested from a partition-scoped WLDF diagnostic system module.
- May not access runtime MBeans for resources in other partitions.
- May configure the Harvester component in a partition-neutral manner. In other words, the `partition-id` or `partition-name` need not be written into in the configuration. This allows WLDF configurations to be portable regardless of the partitions to which they are assigned.

Partition-scoped data that is saved in the Archive component is tagged with partition-specific identifiers so that the data is accessible to system administrators and users of that partition, but not to others.

For a list of the WLST commands that partition administrators can use for diagnostic system modules, see [WLST Diagnostic Commands for Partition Administrators](#).

Configuring Policies and Actions

You can use the WLDF Policies and Actions component within a diagnostic system module to create policies that evaluate common JMX runtime MBean metrics, log records, and instrumentation events. This functionality is available at the partition level within the following constraints:

- Policies may access only those MBean attributes that are available to partitions.
- Log policies are evaluated only for server and domain log records that are specific to the partition.
- SNMP actions are not supported.
- SMTP actions require that the mail session that is used must be visible to the partition in JNDI.
- All policy expressions in partition-scoped diagnostic system modules must use Java Expression Language (EL). See Policy Expression in *Configuring and Using the Diagnostics Framework for Oracle WebLogic Server*.
- Scaling actions (scale up and scale down) are not supported within the scope of a partition.

Accessing Diagnostic Data

As a partition administrator, you do not have file-level access to all log files in the domain, particularly those that are server- or domain-scoped. However, you can selectively access log content pertaining to your partition by using the WLDF Data Accessor and supported WLST functions.

The system administrator, who has full access to all log files in the domain, is responsible for configuring the appropriate security policies for log files to ensure the following:

- You are authenticated when you attempt to access a log record.
- You are able to access log records that are specific to your partition.
- You are denied access to log records not pertaining to your partition.

You can access log records for your partition by using the following MBeans that are created under your partition's [WLDFPartitionRuntimeMBean](#), which is a child of the [PartitionRuntimeMBean](#):

MBean	Description
WLDFPartitionAccessRuntimeMBean	Discovers log files that are available to your partition. Log files that are not available to your partition are not exposed.
WLDFDataAccessRuntimeMBean	Accesses log file records specific to your partition. Log records that do not pertain to your partition are not exposed.

Shared Log Files

The following logs and diagnostic data are shared by all partitions in the domain, and the partition administrator can access records within these logs that pertain to their partition:

Log Type	Content Description
Server	Log events from server and application components pertaining to the partition and that are recorded in the server log file.
Domain	Log events collected centrally from all server instances in the domain pertaining to the partition and that are recorded in the domain log file.
Data source	Data source log events pertaining to the partition and that are recorded in the data source log file.
Harvested data archive	Metric data gathered by the WLDF Harvester component from MBeans pertaining to the partition and that are recorded in the harvested data archive.
Events data archive	WLDF Instrumentation events that are generated by applications deployed in the partition and that are recorded in the events data archive.

When partitions are configured in the domain, each record created in these shared log files includes the following two attributes:

- `partition-name`—a human-readable partition name
- `partition-id`—a unique, automatically generated identifier of the partition

The harvested data archive and events data archive include two additional table columns, `PARTITION_ID` and `PARTITION_NAME`, in which these attributes are listed.

Note:

When partitions are configured in the domain, log records generated on behalf of server- or domain-scoped work are assigned the `partition-id` value 0, and the `partition-name` value `DOMAIN`.

The following example shows a log record entry in a server log file. The `partition-id` and `partition-name` attributes are highlighted in **bold**.

```
####<Oct 2, 2015 1:20:28 PM EDT> <Notice> <Partition Lifecycle>
<tiger-mac.local> <partitionAdmin> <[ACTIVE] ExecuteThread: '2' for queue:
'weblogic.kernel.Default (self-tuning)'> <system> <>
<cfd4d584-ee45-49a6-ae91-9c12551330d8-000000af> <1443806428565> <[severity-value:
32] [rid: 0]
[partition-id: d75a4899-ca61-4ed8-b519-317b0ec15061] [partition-name: p1] >
```

```
<BEA-2192303> <The partition lifecycle operation "START" for partition "p1" is initiated.>
```

When the WLDF Data Accessor obtains log records pertaining to a specific partition, it filters them based on the `partition-id` value.

 **Note:**

The Java Flight Recorder (JFR) flight recording, which includes events from the JVM and from any other event producer, such as WebLogic Server and Dynamic Monitoring Service (DMS), can include `partition-id` and `partition-name` attributes to distinguish data among partitions. However, only the system administrator may have access to the JFR data that contains the partition information that can be used to diagnose multitenancy issues.

Partition-Specific Log Files

The following log files are specific to each partition and are available to the partition administrator. These log files are stored in the `partitions/<partition>/system/servers/<server-name>` directory.

Log Type	Content Description
HTTP access log	Log events of all HTTP transactions from the partition's virtual target web server, which are stored in the file <code>access.log</code> .
JMS server	JMS server message lifecycle events for JMS server resources that are defined within a resource group, or resource group template, and that are scoped to the partition.
SAF agent	SAF agent message lifecycle events for SAF agent resources that are defined within a resource group, or resource group template, and that are scoped to the partition.
Connector	Log data generated by Java EE Connector Architecture resource adapter modules that are deployed to a resource group, or resource group template, within the partition.
Servlet context	Servlet context log data generated by Java EE web application modules that are deployed to a resource group, or resource group template, within the partition.

Using the WLDF Data Accessor

Using the WLDF Data Accessor, you can perform data lookups by type, component, and attribute. You can filter by severity, source, and content. You can also access diagnostic data in tabular form. For more information about the Data Accessor, see *Accessing Diagnostic Data With the Data Accessor* in *Configuring and Using the Diagnostics Framework for Oracle WebLogic Server*.

The following table lists the logical names for each of the log and diagnostic data files you can view using the Data Accessor. You use the logical name as a key to refer to a log type when using the `WLDFDataAccessRuntimeMBean` or `WLST`.

Log Type	Logical Name
Server log	ServerLog
Domain log	DomainLog
JDBC log	DataSourceLog
Harvested data archive	HarvestedDataArchive
Events data archive	EventsDataArchive
HTTP access log	HTTPAccessLog/<WebServer-Name>
JMS server	JMSMessageLog/<JMSServer-Name>
SAF agent	JMSSAFMessageLog/<SAFAgent-Name>
Servlet context	WebAppLog/<WebServer-Name>/context-path
Connector	ConnectorLog/connection-Factory-jndiName\$partition-name

As a partition administrator, you can use the following tools to access log and diagnostic data for your partition:

- **Fusion Middleware Control**
You can use Fusion Middleware Control to view records from both shared and partition-specific log files. See Monitor domain partitions in *Administering Oracle WebLogic Server with Fusion Middleware Control*.
- **WebLogic Scripting Tool**
For a list of the WLST commands that partition administrators can use for using the Data Accessor, see [WLST Diagnostic Commands for Partition Administrators](#). For the syntax and examples of these commands, see Diagnostics Commands in *WLST Command Reference for WebLogic Server*.

The Oracle WebLogic Server Administration Console does not support viewing partition log and diagnostic data.

Monitoring Resource Consumption Management

Resource consumption management (RCM) provides a flexible, dynamic mechanism for system administrators to manage shared resources and provide consistent performance of domain partitions in MT environments. WebLogic Server supports the ability to monitor RCM within a partition scope.

For more information about RCM, see [Configuring Resource Consumption Management](#).

Configuring RCM

For information about configuring RCM, see [Configuring Resource Consumption Management: Main Steps](#).

Partition-Scoped RCM Metrics

[Table 26-1](#) lists and summarizes the attributes on the `PartitionResourceMetricsRuntimeMBean` that can provide partition-scoped RCM metrics:

Table 26-1 PartitionResourceMetricsRuntimeMBean Attributes for RCM

Attribute	Description
<code>RCMMetricsDataAvailable</code>	Boolean value indicating whether RCM metrics data is available for this partition.
<code>CpuTimeNanos</code>	Total CPU time spent in the context of a partition in the time since server start or partition creation, whichever is later.
<code>AllocatedMemory</code>	Total allocated memory in bytes in the context of a partition in the time since server or partition creation, whichever is later.
<code>RetainedHeapHistoricalData</code>	Snapshot of the historical data for retained heap memory usage for the partition. Data is returned as a two-dimensional array for the usage of retained heap scoped to the partition over time. Each item in the array contains a tuple of [<code>timestamp (long)</code> , <code>retainedHeap(long)</code>] values.
<code>CpuUtilizationHistoricalData</code>	Snapshot of the historical data for CPU usage for the partition. CPU use percentage indicates the percentage of CPU used by a partition with respect to available CPU to Weblogic Server. Data is returned as a two-dimensional array for the CPU usage scoped to the partition over time. Each item in the array contains a tuple of [<code>timestamp (long)</code> , <code>cpuUsage(long)</code>] values.
<code>ThreadCount</code>	Number of threads currently assigned to the partition.
<code>TotalOpenedSocketCount</code>	Total number of sockets opened in the context of a partition in the time since server start or partition creation, whichever is later.
<code>CurrentOpenSocketCount</code>	Number of sockets currently open in the context of a partition.
<code>NetworkBytesRead</code>	Total number of bytes read from sockets for a partition in the time since server start or partition creation, whichever is later.

Table 26-1 (Cont.) PartitionResourceMetricsRuntimeMBean Attributes for RCM

Attribute	Description
NetworkBytesWritten	Total number of bytes written to sockets for a partition in the time since server start or partition creation, whichever is later.
TotalOpenedFileCount	Total number of files opened in the context of a partition in the time since server start or partition creation, whichever is later.
CurrentOpenFileCount	Number of files currently open in the context of a partition.
FileBytesRead	Total number of bytes read in the context of a partition in the time since server start or partition creation, whichever is later.
FileBytesWritten	Total number of bytes written in the context of a partition in the time since server start or partition creation, whichever is later.
TotalOpenedFileDescriptorCount	Total number of file descriptors opened in the context of a partition in the time since server start or partition creation, whichever is later.
CurrentOpenFileDescriptorCount	Number of file descriptors currently open in the context of a partition.

For general RCM configuration requirements, see [Configuring Resource Consumption Management](#).

Instrumenting Partition-Scoped Applications

The WLDF Instrumentation component provides a mechanism for adding diagnostic code to WebLogic Server instances and the applications running on them.

For more information about this feature, see *Configuring Instrumentation in Configuring and Using the Diagnostics Framework for Oracle WebLogic Server*.

WebLogic Server MT supports the ability to instrument an application that is deployed within a partition by using the deployment descriptor (`META-INF/weblogic-diagnostics.xml`) or a deployment plan. This capability is supported with the following conditions:

- The system administrator must configure the diagnostic system module at the system level and target it to the appropriate Managed Server instance at the global level (that is, not partition-scoped), and its instrumentation component must be enabled.
- The instrumented application must contain the `META-INF/weblogic-diagnostics.xml` deployment descriptor and its Instrumentation component must be enabled.
- The application must not be configured in such a way that it shares classloaders with application instances in other partitions.

Configuring Partition-Scoped Diagnostic Image Capture

Diagnostic image capture contains relevant data from various subsystems of WebLogic Server as it exists when a diagnostic image capture is initiated. When a partition user initiates a diagnostic image capture, the contents of the image are limited to the partition-scoped resources only.

When a diagnostic image capture is initiated by calling an operation on the [WLDFPartitionImageRuntimeMBean](#) that corresponds to a partition, the generated images contain data pertaining to that partition only. For partition-scoped diagnostic images, the `partition-id` and `partition-name` attributes are encoded into the file names.

Note that only the system administrator may have access to the JFR data containing the information corresponding to a partition.

You can initiate diagnostic image capture manually with any of the following tools:

- WebLogic Server Administration Console
- Fusion Middleware Control
- WLST
- JMX
- WLDF Policies and Actions component

When scoped to a partition, the following components can generate partition-specific records into the diagnostic image capture:

- Connector
- Instrumentation
- JDBC
- JNDI
- JVM
- Logging
- RCM
- Work Manager
- JTA

For a list of the WLST commands that partition administrators can use for diagnostic image capture, see [WLST Diagnostic Commands for Partition Administrators](#).

WLST Diagnostic Commands for Partition Administrators

WLST diagnostics commands are available to partition administrators for accessing partition-scoped logging and diagnostics data.

These commands are presented in three categories:

- **Data Accessor:** Commands for using the WLDF Data Accessor component to access partition-scoped diagnostic data from various sources, including log records, data events, and harvested metrics

- Diagnostic image capture: Commands for capturing, saving, and accessing diagnostic image capture
- Diagnostic modules: Commands for exporting metric data collected by a diagnostic system module within a specific time interval

Each of these commands includes a *partition* argument for specifying the partition name.

Command	Category	Summary
<code>exportDiagnosticDataFromServer</code>	Data Accessor	Executes a query on the server side and retrieves the exported WLDF data.
<code>getAvailableDiagnosticDataAccessorsNames</code>	Data Accessor	Gets the logical names of diagnostic Data Accessor instances currently available on a server or partition, and returns them as an array of string values.
<code>captureAndSaveDiagnosticImage</code>	Diagnostic image	Captures a diagnostic image and downloads it to the client.
<code>getAvailableCapturedImages</code>	Diagnostic image	Returns, as an array of strings, a list of the previously captured diagnostic images that are stored in the image destination directory configured on the server.
<code>saveDiagnosticImageCaptureEntryFile</code>	Diagnostic image	Downloads a specific entry from the diagnostic image capture that is located on the server to which WLST is currently connected.
<code>saveDiagnosticImageCaptureFile</code>	Diagnostic image	Downloads the specified diagnostic image capture from the server to which WLST is currently connected.
<code>purgeCapturedImages</code>	Diagnostic image	Purges the diagnostic image files on the server as per the age criteria specified from the server's configured image destination directory.
<code>exportHarvestedTimeSeriesData</code>	Diagnostic module	Exports harvested metric data from the diagnostic archive file for a particular server-scoped or partition-scoped diagnostic system module.

For more information about the WLST commands available for diagnostics, see *Diagnostics Commands in WLST Command Reference for WebLogic Server*.