

# **Oracle WebLogic Server 11g: Advanced Administration**

**Volume I • Student Guide**

D58686GC20

Edition 2.0

December 2010

D71658

**ORACLE®**

**Author**

TJ Palazzolo

**Technical Contributors  
and Reviewers**

Prashant Agarwal

Tom Barnes

Steve Button

Dave Cabelus

Josh Dorr

Arvind Jain

Anthony Lai

Serge Moiseev

Craig Perez

Mark Prichard

Sandeep Shrivastava

John Van Pelt

**Editor**

Malavika Jinka

**Graphic Designer**

Maheshwari Krishnamurthy

**Publishers**

Revathi Ramamoorthy

Srividya Rameshkumar

**Copyright © 2010, Oracle and/or its affiliates. All rights reserved.**

**Disclaimer**

This document contains proprietary information and is protected by copyright and other intellectual property laws. You may copy and print this document solely for your own use in an Oracle training course. The document may not be modified or altered in any way. Except where your use constitutes "fair use" under copyright law, you may not use, share, download, upload, copy, print, display, perform, reproduce, publish, license, post, transmit, or distribute this document in whole or in part without the express authorization of Oracle.

The information contained in this document is subject to change without notice. If you find any problems in the document, please report them in writing to: Oracle University, 500 Oracle Parkway, Redwood Shores, California 94065 USA. This document is not warranted to be error-free.

**Restricted Rights Notice**

If this documentation is delivered to the United States Government or anyone using the documentation on behalf of the United States Government, the following notice is applicable:

**U.S. GOVERNMENT RIGHTS**

The U.S. Government's rights to use, modify, reproduce, release, perform, display, or disclose these training materials are restricted by the terms of the applicable Oracle license agreement and/or the applicable U.S. Government contract.

**Trademark Notice**

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

## Contents

### 1 Course Overview

- Objectives 1-2
- Target Audience 1-3
- Introductions 1-4
- Course Schedule 1-5
- Course Practices 1-7
- Classroom Guidelines 1-8
- For More Information 1-9
- Oracle by Example (OBE) 1-10
- Related Training 1-11

### 2 Installation Management

- Objectives 2-2
- WebLogic Server Review 2-3
- WebLogic Smart Update 2-4
- WebLogic Patch Terminology 2-5
- Getting Started 2-6
- Downloading and Applying Patches 2-7
- WebLogic Start Script Review 2-8
- Patch Classpath 2-9
- Start Script Editor 2-10
- Smart Update Command-Line Interface 2-11
- Patch Distribution 2-12
- Oracle Cloning Tool 2-13
- Quiz 2-14
- Summary 2-16

### 3 Domain Templates

- Objectives 3-2
- Road Map 3-3
- Domain Template Review 3-4
- Template Contents 3-5
- Template Exclusions 3-6
- Script Replacement Variables 3-7
- Template SQL Scripts 3-8

Extension Template Concepts	3-10
Fusion Middleware (FMW) Templates	3-11
Section Summary	3-12
Road Map	3-13
Creating a Domain Template	3-14
Selecting the Template Domain Source	3-15
Selecting the Template Destination: Adding Files to the Template	3-16
Adding SQL Scripts to the Template	3-17
Configuring the Administration Server	3-18
Defining Users, Groups, and Roles	3-19
Configuring Group Membership	3-20
Configuring Role Membership	3-21
Creating Windows Start Menu Entries	3-22
Preparing Scripts and Files with Variables	3-23
Creating an Extension Template	3-24
Section Summary	3-26
Quiz	3-27
Lesson Summary	3-29
Practice 3-1: Create a Custom Domain Template	3-30

#### **4 Other Domain Tools**

Objectives	4-2
Review of Basic WLST Commands	4-3
WLST Best Practice: Variable Declaration	4-4
WLST Best Practice: Password Management	4-5
WLST Best Practice: Error Handling	4-6
Working with Templates Using WLST	4-7
Template WLST Examples	4-8
Pack Tool	4-9
Template Types	4-10
Creating Managed Server Templates	4-11
Unpack Tool	4-13
ConfigToScript	4-14
Generated Script Internals	4-15
Domain Configuration Archives	4-16
WebLogic Auditing Provider	4-17
Security Audit Events	4-18
Adding an Auditing Provider	4-19
Configuring the Auditing Provider	4-20
Configuration Audit Events	4-21
Quiz	4-22

Summary 4-24

Practice 4-1: Work with Templates from the Command Line 4-25

## 5 Advanced Network Configuration

Objectives 5-2

Default WebLogic Networking 5-3

Additional Networking Scenarios 5-5

Dedicate Network Interfaces to Specific Servers 5-6

Use Multiple Ports on a Single Server 5-7

Isolate Administrative Communication 5-8

Isolate Cluster Communication 5-9

Network Channels 5-10

Channel Selection 5-11

Creating a Channel 5-12

Channel Network Settings 5-14

Channel WLST Example 5-15

Monitoring Channels 5-16

Creating Administration Channels 5-17

Administration Port 5-18

Administration Port Override 5-19

Server Standby Mode 5-20

Quiz 5-21

Summary 5-23

Practice 5-1: Use Network Channels 5-24

## 6 Multi Data Sources

Objectives 6-2

Data Source Review 6-3

XA Data Source Review 6-4

Multi Data Sources 6-5

Multi Data Source Architecture 6-6

Failover Option 6-7

Load Balancing Option 6-8

Connection Testing 6-9

Creating a Multi Data Source 6-10

Configuring a Multi Data Source 6-12

Multi Data Source WLST Example 6-13

Managing Multi Data Source Members 6-14

Oracle Real Application Clusters (RAC) Overview 6-15

Oracle GridLink for RAC 6-16

RAC Services 6-17

WLS and RAC Services	6-18
Creating Data Sources to Support RAC Services	6-20
Additional RAC Considerations	6-21
Quiz	6-23
Summary	6-25
Practice 6-1: Use a Multi Data Source for Failover	6-26

## **7 JDBC Performance Essentials**

Objectives	7-2
JDBC and Application Design	7-3
Connection Pooling	7-4
Statement Caching	7-5
Connection Pinned to Thread	7-6
Logging Last Resource (LLR) Transactions	7-7
LLR Example	7-8
Configuring LLR	7-9
Quiz	7-10
Summary	7-12

## **8 JMS Message Management**

Objectives	8-2
WebLogic JMS Review	8-3
Destination Management Features	8-5
Viewing Messages	8-6
Message Contents	8-8
Message States	8-9
Publishing a Test Message	8-10
Moving Messages	8-11
Exporting Messages to XML	8-12
Exported Messages Format	8-13
Pausing Destinations	8-14
Pausing a Destination	8-15
JMS Management WLST Examples	8-16
JMS Logging	8-17
Configuring Destination Logging	8-18
Quiz	8-19
Summary	8-21

## **9 JMS Guaranteed Messaging**

Objectives	9-2
Guaranteed Messaging Concepts	9-3

Persistent Messaging	9-5
Default Persistent Store	9-6
Comparing File and JDBC Stores	9-7
Creating a File Store	9-8
Creating a JDBC Store	9-9
JDBC Store Details	9-10
Assigning a Store to a JMS Server	9-11
Persistent Store WLST Example	9-12
Connection Factory and Destination Persistence	9-13
Topics and Durable Subscribers	9-14
Connection Factory Client ID	9-15
Monitoring Durable Subscribers	9-16
Message Paging	9-17
Configuring Message Paging	9-18
Quiz	9-19
Summary	9-21
Practice 9-1: Configure JMS Persistence	9-22

## **10 JMS Performance Essentials**

Objectives	10-2
JMS and Application Design	10-3
JMS Quotas Review	10-5
Configuring a JMS Server Quota	10-6
Creating a Destination Quota	10-7
Send Timeout	10-8
Quota Blocking Policies	10-9
Thresholds and Flow Control Review	10-10
Configuring Thresholds	10-11
Message Compression	10-12
Scan Expiration Interval	10-13
Message Ordering	10-14
Unit of Order (UOO)	10-15
Connection Factory UOO	10-16
Quiz	10-17
Summary	10-19

## **11 JMS Store and Forward**

Objectives	11-2
WebLogic Store and Forward (SAF)	11-3
SAF Architecture	11-5
Creating an SAF Agent	11-6

Configuring an SAF Agent	11-7
Creating a Remote SAF Context	11-9
Creating an SAF Error Handler	11-10
SAF Imported Destinations	11-11
Creating SAF Imported Destinations	11-12
Adding Remote Endpoints	11-14
Configuring Remote Endpoints	11-15
Store and Forward WLST Example	11-16
Managing SAF Agents	11-17
Managing an SAF Agent	11-18
Managing SAF Agent Endpoints	11-19
Web Services Reliable Messaging (WSRM)	11-20
Quiz	11-22
Summary	11-24
Practice 11-1: Store and Forward JMS Messages	11-25

## **12 JMS Message Bridge**

Objectives	12-2
WebLogic Server Messaging Bridge	12-3
Messaging Bridge Architecture	12-4
Bridging Scenarios	12-5
Comparing Bridges to Store and Forward	12-6
Resource Adapter Review	12-7
Bridge Adapters	12-8
JMS Bridge Destinations	12-10
Creating a JMS Bridge Destination	12-11
Creating a Message Bridge	12-13
Configuring a Message Bridge	12-16
Message Bridge WLST Example	12-18
Oracle Advanced Queuing (AQ) Overview	12-19
AQ Integration Overview	12-20
Creating a Foreign Server for AQ	12-21
Quiz	12-22
Summary	12-24
Practice 12-1: Bridge JMS Providers	12-25

## **13 Server Migration**

Objectives	13-2
WebLogic Clustering Review	13-3
Whole-Server Migration	13-4

Node Manager Review	13-5
Leasing Service	13-7
Automatic Server Migration Architecture: No Failure	13-8
Automatic Server Migration Architecture: Machine Failure	13-9
Leasing Types	13-10
Configuration Overview	13-11
Network Considerations	13-12
Node Manager Network Configuration	13-13
Configuring Cluster Leasing	13-14
Database Leasing Schema	13-15
Enabling Automatic Migration for a Server	13-16
Candidate Machines	13-17
Machine Failback	13-18
Manual Server Migration	13-19
Migrating a Server	13-20
Additional File System Considerations	13-21
WebLogic Operations Control (WLOC) Overview	13-22
WLOC Architecture	13-23
Quiz	13-24
Summary	13-26
Practice 13-1: Migrate Failed Servers	13-27

## **14 JMS Clustering**

Objectives	14-2
Road Map	14-3
JMS Communication Review	14-4
Clustered JNDI Access	14-6
Pinned Resources	14-7
JMS Cluster Targeting	14-8
Clustered Connection Factories	14-9
Reconnect Policy	14-10
Service Migration	14-11
Migratable Targets	14-12
Default Migratable Targets	14-14
Service Migration Policies for Migratable Targets	14-15
Configuring Migratable Targets	14-16
Assigning Resources to Migratable Targets	14-18
Automatic Transaction Migration	14-19
Migrating Services Manually	14-20
Service Migration WLST Example	14-21
Pre/Post Migration Scripts	14-22

Section Summary	14-24
Practice 14-1: Configure JMS High Availability	14-25
Road Map	14-26
JMS and Scalability	14-27
Distributed Destination Architecture	14-28
Distributed Queues and Topics	14-29
Creating a Distributed Destination	14-30
Targeting a Distributed Destination	14-31
Distributed Destinations and the Configuration Wizard	14-32
Distributed Destination WLST Example	14-33
Default Load Balancing Constraints	14-34
Connection Factories and Distributed Destinations	14-36
Message-Driven Bean (MDB) Review	14-37
MDBs and Distributed Destinations	14-38
Section Summary	14-39
Quiz	14-40
Lesson Summary	14-42
Practice 14-2: Load Balance JMS Messages	14-43

## **15 Cross-Cluster Replication**

Objectives	15-2
WebLogic Session Persistence Review	15-3
In-Memory Replication Review	15-4
Using Multiple Clusters	15-6
Cross-Cluster Replication	15-7
Cross-Domain Security	15-8
Configuring Cross-Domain Security	15-9
Metropolitan Area Network (MAN) Replication	15-10
MAN Replication Example	15-11
MAN Replication Example: Server Failover	15-12
Wide Area Network (WAN) Replication	15-13
WAN Replication Example	15-14
WAN Replication Example: Cluster Failover	15-15
Configuring Cross-Cluster Replication	15-16
Advanced MAN and WAN Settings	15-17
WAN Replication Schema	15-18
Oracle Coherence Overview	15-19
Coherence*Web Overview	15-21
Coherence*Web and WebLogic Clusters	15-22
Coherence*Web Configuration Overview	15-23
Quiz	15-24

Summary 15-26  
Practice 15-1: Replicate Sessions Across Two Clusters 15-27

## 16 Authentication Providers

Objectives 16-2  
Road Map 16-3  
Security Realm Review 16-4  
Security Provider Stores 16-6  
Store Implementations 16-7  
Configuring the RDBMS Store by Using the Configuration Wizard 16-8  
Default Security Configuration 16-9  
Security Customization Approaches 16-10  
Creating a New Security Realm 16-11  
Activating a Security Realm 16-12  
Security Realm WLST Example 16-13  
Authentication Provider Review 16-14  
Available Authentication Providers 16-15  
Multiple Authentication Providers 16-17  
Control Flags 16-18  
Administration Groups 16-19  
Section Summary 16-20  
Road Map 16-21  
Lightweight Directory Access Protocol (LDAP) 16-22  
LDAP Structure 16-23  
LDAP Search Operations 16-24  
LDAP Authentication Providers 16-25  
Available LDAP Authentication Providers 16-26  
Configuring an LDAP Provider: Connection 16-27  
Configuring an LDAP Provider: Users 16-28  
Configuring an LDAP Provider: Groups 16-29  
Configuring an LDAP Provider: Subgroups 16-30  
Configuring an LDAP Provider: Dynamic Groups 16-31  
LDAP Provider WLST Example 16-32  
LDAP Failover 16-33  
LDAP Caching 16-34  
LDAP X509 Identity Asserter Overview 16-35  
Section Summary 16-37  
Practice 16-1: Authenticate Using an External LDAP 16-38  
Road Map 16-39  
Database Authentication Providers 16-40  
Available Database Authenticators 16-41

Authenticator Data Sources	16-42
Configuring the SQL Authenticator	16-43
Configuring the SQL Authenticator: Passwords	16-44
Configuring the SQL Authenticator: Subgroups	16-45
SQL Authenticator WLST Example	16-46
Default Schema	16-47
Section Summary	16-48
Practice 16-2: Authenticate Using a Database	16-49
Road Map	16-50
Password Validation Providers	16-51
Authentication Provider Support	16-52
Default Authenticator and Password Validation	16-53
Password Composition Rules	16-54
Accessing the Password Validator	16-55
Configuring the Password Validator	16-56
Password Validator WLST Example	16-58
Section Summary	16-59
Practice 16-3: Define Password Rules	16-60
Road Map	16-61
Security Migration Scenarios	16-62
Provider Migration Support	16-63
Migration Approaches	16-64
Migration Formats	16-65
Migration Example: XACML Authorizer	16-66
Security Migration WLST Example	16-67
Section Summary	16-68
Quiz	16-69
Lesson Summary	16-72

## **17 Server Performance Essentials**

Objectives	17-2
Road Map	17-3
Performance Terminology	17-4
Bottlenecks	17-5
Benchmarking	17-6
Performance Testing Methodology	17-7
Load Testing	17-8
Load Testing Tools	17-9
The Grinder	17-10
The Grinder Architecture	17-11
The Grinder Proxy	17-12

Agent Properties	17-13
The Grinder Console	17-14
Section Summary	17-15
Road Map	17-16
Java Virtual Machine (JVM) Selection	17-17
Setting WLS JVM Arguments	17-18
JRockit Review	17-19
JVM Tuning Parameters	17-20
Java Heap	17-21
WLS Production Recommendations for Any JVM	17-22
Generational Garbage Collection	17-23
Basic JRockit JVM Arguments	17-25
JRockit Recommendations for WLS	17-27
Sun Hotspot JVM Generational GC	17-28
Basic Sun JVM Arguments	17-29
Sun JVM Recommendations for WLS	17-30
Monitoring a WLS JVM	17-31
WLS Low Memory Detection	17-32
Configuring Low Memory Detection	17-33
JRockit Command-Line Heap Monitoring	17-34
Sun JVM Heap Monitoring	17-35
JRockit Mission Control (JRMC)	17-36
Management Console Features	17-37
Management Console Heap Monitoring	17-38
Flight Recorder Overview	17-39
Flight Recorder Features	17-40
Flight Recorder Code Analysis	17-41
Section Summary	17-42
Practice 17-1: Tune a Server JVM	17-43
Road Map	17-44
Production Mode	17-45
Memory Chunk Settings	17-46
Logging Considerations	17-47
Log Filters Review	17-48
JavaServer Page (JSP) Review	17-49
Precompiling JSP Files	17-50
JSP and Class Reloading	17-51
Using Web Servers for Static Content	17-52
Cluster Considerations	17-53
Session Persistence Cache Size	17-54
Section Summary	17-55

Practice 17-2: Tune Server Performance	17-56
Road Map	17-57
WebLogic Server Threads	17-58
Monitoring a Server Thread Pool	17-59
Monitoring Server Threads	17-60
Stuck Thread Handling	17-61
Configuring Stuck Thread Handling	17-62
Application Stuck Thread Handling	17-64
Work Managers	17-65
Work Manager Scope	17-66
Work Manager Architecture	17-67
Request Class Types	17-68
Creating a Work Manager	17-69
Creating a Request Class	17-70
Constraint Types	17-71
Creating a Constraint	17-72
Work Manager WLST Example	17-73
Work Managers and Stuck Threads	17-74
Assigning Work Managers to Applications	17-75
Section Summary	17-76
Quiz	17-77
Lesson Summary	17-80
Practice 17-3: Tune Performance Using Work Managers	17-81

## **18 Monitoring and Diagnostics Essentials**

Objectives	18-2
Road Map	18-3
Why Monitoring Tools?	18-4
Oracle Monitoring Tools	18-5
Java Management Extension (JMX) Review	18-6
WLS MBean Hierarchies	18-7
Monitoring with the Console	18-8
Monitoring with WLST	18-9
Runtime MBean Documentation	18-10
Fusion Middleware (FMW) Control Review	18-11
FMW Control: Viewing Farm Topology	18-12
FMW Control: Monitoring Servers	18-13
FMW Control: Monitoring Deployments	18-14
Guardian Overview	18-15
Section Summary	18-16
Road Map	18-17

WebLogic Diagnostics Framework (WLDF) 18-18
WLDF Architecture 18-19
WLDF Configuration Overview 18-20
Capturing a Server Diagnostic Image 18-21
Diagnostic Archives 18-23
Configuring Server Diagnostic Archives 18-24
Creating a Diagnostic Module 18-25
Metric Collectors 18-26
Configuring a Metric Collector 18-27
Watches and Notifications 18-28
Configuring a Watch 18-29
WLDF WLST Examples 18-31
Sample WLDF Framework 18-32
Instrumentation 18-34
Working with Diagnostic Monitors 18-36
Configuring a System-Scoped Monitor 18-37
WLDF Monitoring Dashboard 18-39
Views, Charts, and Graphs 18-40
Anatomy of a Chart 18-41
Creating a New Graph 18-42
Section Summary 18-43
Practice 18-1 Configure and Monitor Diagnostic Data 18-44
Road Map 18-45
Simple Network Management Protocol (SNMP) 18-46
SNMP Architecture 18-47
Object Identifier (OID) 18-48
Management Information Base (MIB) 18-49
WLS MIB and OIDs 18-50
Common SNMP Message Types 18-51
WLS SNMP Architecture 18-52
Creating an SNMP Agent 18-54
Configuring an SNMP Agent 18-55
SNMP Channels 18-57
WLS SNMP Notifications 18-58
Creating Trap Monitors 18-59
Creating Trap Destinations 18-60
SNMP Security 18-61
Configuring Agent Security 18-62
Configuring SNMPv3 Credentials 18-63
Configuring Trap Destination Security 18-64
WLS SNMP Utility 18-65

Section Summary	18-66
Practice 18-2 Monitor WLS Using SNMP	18-67
Road Map	18-68
Subsystem Debugging	18-69
Debug Scopes	18-70
Example Debug Scopes and Attributes	18-71
Debug Logging	18-73
Section Summary	18-74
Quiz	18-75
Lesson Summary	18-79
Practice 18-3 Debug WLS Subsystems	18-80

# 1

## Course Overview

ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

# Objectives

After completing this course, you will be able to:

- Create custom domain templates
- Configure server network channels
- Define a data source for load balancing or failover
- Configure Java Message Service (JMS) store and forward and bridging features
- Configure JMS load balancing and failover in a cluster
- Configure automatic server migration in a cluster
- Integrate WebLogic Server with an external Lightweight Directory Access Protocol or database security store
- Tune a Java Virtual Machine and WebLogic Server subsystems for performance
- Configure simple diagnostic collectors and notifications



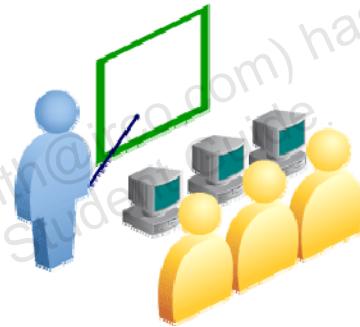
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Target Audience

This course is for experienced WebLogic Server administrators or those who have completed *WebLogic Server: Administration Essentials*.

Prerequisite skills include:

- Basic administration console navigation
- Basic WLST commands
- Basic JDBC and JMS configuration
- Application deployment



ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Target Audience

If you are concerned about whether your experience fulfills the course prerequisites, ask the instructor.

# Introductions

Introduce yourself.

Tell us about:

- Your company and role
- Your experience with WebLogic Server
- Any previous Oracle product experience



ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

# Course Schedule

Day		Lesson
1	AM	Course Overview Installation Management Domain Templates
	PM	Other Domain Tools Advanced Network Configuration Multi Data Sources JDBC Performance Essentials
2	AM	JMS Message Management JMS Guaranteed Messaging JMS Performance Essentials
	PM	JMS Store and Forward JMS Message Bridge Server Migration



ORACLE

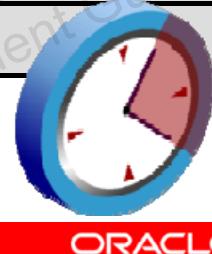
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Course Schedule

The class schedule might vary according to the pace of the class. The instructor will provide updates.

# Course Schedule

Day		Lesson
3	AM	JMS Clustering
	PM	Cross-Cluster Replication Authentication Providers
4	AM	Authentication Providers <i>continued</i> Server Performance Essentials
	PM	Server Performance Essentials <i>continued</i>
5	AM	Monitoring and Diagnostics Essentials
	PM	Monitoring and Diagnostics Essentials <i>continued</i>



## Course Practices

- Each topic will be reinforced with a hands-on exercise.
- Most exercises include a scripted solution to aid the students who fall behind.



ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Classroom Guidelines

- The instructor starts each session at the scheduled time.
- Do ask questions, but be respectful of the topic at hand and the interest of other students.
- Ensure that cell phones and pagers are silent.



ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Classroom Guidelines

We hope that these guidelines will help the class proceed smoothly and enable you to get the maximum benefit from the course.

## For More Information

Topic	Website
<b>Education</b>	<a href="http://education.oracle.com">http://education.oracle.com</a>
<b>Product Documentation</b>	<a href="http://www.oracle.com/technology/documentation">http://www.oracle.com/technology/documentation</a>
<b>Product Downloads</b>	<a href="http://www.oracle.com/technetwork/indexes/downloads">http://www.oracle.com/technetwork/indexes/downloads</a>
<b>Product Articles</b>	<a href="http://www.oracle.com/technetwork/articles">http://www.oracle.com/technetwork/articles</a>
<b>Product Support</b>	<a href="http://www.oracle.com/support">http://www.oracle.com/support</a>
<b>Product Forums</b>	<a href="http://forums.oracle.com">http://forums.oracle.com</a>
<b>Product Tutorials/Demos</b>	<a href="http://www.oracle.com/technetwork/tutorials/index.html">http://www.oracle.com/technetwork/tutorials/index.html</a>
<b>Sample Code</b>	<a href="https://www.samplecode.oracle.com/">https://www.samplecode.oracle.com/</a>



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## For More Information

This course and the instructor will attempt to address any questions that you might pose, but after you complete the course, Oracle provides a variety of channels for developers and administrators to access additional information.

## Oracle by Example (OBE)

OBEs are free online step-by-step tutorials that cover specific product features.

### Oracle by Example Series: Oracle WebLogic Server 10.3

The Oracle by Example (OBE) series provides step-by-step instructions on how to perform a Server 10.3. The Oracle by Example series reduces the time spent investigating what steps the step-by-step solutions are built for practical real world situations, not only is knowledge gained experience, but also the solutions presented may then be used as the foundation for production time to deployment.

To view a lesson, click on the link below.

#### List of Lessons

##### Installation and Configuration

- [Installing and Configuring Oracle WebLogic Server Instance](#)
- [Configuring Oracle WebLogic Server Managed Instances](#)
- [Managing Machines using the Administration Console](#)



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Oracle by Example (OBE)

The Oracle by Example (OBE) series provides hands-on, step-by-step instructions on how to implement various technology solutions to business problems. OBE solutions are built for practical real-world situations, allowing you to gain valuable hands-on experience as well as use the presented solutions as the foundation for production implementation, dramatically reducing time to deployment.

## Related Training

Course
<i>WebLogic Server: Monitor and Tune Performance</i>
<i>WebLogic Server: Diagnostics and Troubleshooting</i>
<i>Introduction to WebLogic Server for OC4J Administrators</i>
<i>WebLogic Server: Overview for OC4J Administrators</i>



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Related Training

Your instructor can provide additional information regarding the availability and contents of the courses listed above.

Unauthorized reproduction or distribution prohibited. Copyright© 2011, Oracle and/or its affiliates.

LakshmiKanth Kemburaj (c-klakshmikanth@irc0.com) has a  
non-transferable license to use this Student Guide.

## Installation Management

ORACLE®

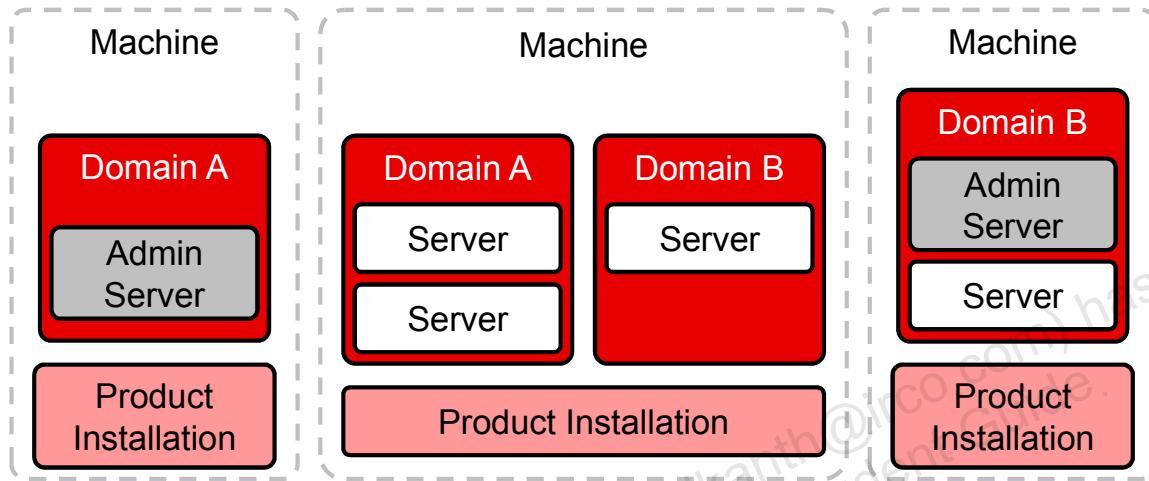
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

# Objectives

After completing this lesson, you will be able to:

- Describe the capabilities of the Smart Update tool
- Download and apply a product patch
- Explain the relationship between patches and server classpaths

# WebLogic Server Review



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## WebLogic Server Review

An Oracle WebLogic Server administration domain is a logically related group of Oracle WebLogic Server resources. Domains include a special Oracle WebLogic Server instance called the Administration Server, which is the central point from which you configure and manage all resources in the domain. Usually, you configure a domain to include additional Oracle WebLogic Server instances called Managed Servers. You deploy Web applications, EJBs, Web Services, and other resources onto the Managed Servers and use the Administration Server for configuration and management purposes only.

You can use a single Oracle WebLogic Server installation to create and run multiple domains, or you can use multiple installations to run a single domain. How you organize your Oracle WebLogic Server installations into domains depends on your business needs. You can define multiple domains based on different system administrators' responsibilities, application boundaries, or geographical locations of the machines on which servers run. Conversely, you might decide to use a single domain to centralize all Oracle WebLogic Server administration activities.

For development or test environments, you can create a simple domain that consists of a single server instance. This single instance acts as an Administration Server and hosts the applications that you develop.

# WebLogic Smart Update

## Smart Update:

- Is a tool included with WebLogic Server
- Connects to the Oracle Support network or your local WebLogic Server patch repository
- Validates your WebLogic Server installation against the latest list of product patches
- Downloads patches and applies them to your product installation
- Can remove patches previously installed by Smart Update
- Performs conflict checking



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## WebLogic Smart Update

Smart Update is a stand-alone Java application that you can run to upgrade your WebLogic Server installations quickly and easily with maintenance patches and maintenance packs. Smart Update supports a model in which patches, which have been downloaded to a central location in an organization, can be distributed via script to machines in a production environment on which products have been installed. This enables you to adapt Smart Update to your business practices that govern how maintenance updates are distributed and applied to machines, especially those in a production environment from which a direct connection to Oracle Support for downloading patches is not appropriate or possible.

Each patch created for a particular maintenance level of a product is validated against all existing patches for that maintenance level. For example, when you attempt to apply one patch that depends on another that has not been applied, Smart Update notifies you of the dependency. You can then download the prerequisite patch and apply it before proceeding. Patch validation is performed automatically whenever you apply a patch. You also have the option of requesting patch validation before downloading a patch.

Use the Smart Update tool to view, download, and apply available patches. If a patch is not available from Smart Update to fix your problem, open a case with Oracle Support.

# WebLogic Patch Terminology

Patches can include native libraries, Java classes/libraries, Web server plug-ins, and/or product configuration files.

Term	Definition
<b>Maintenance Pack</b>	A collection of patches that are formally released and bundled together as a single installer with an incremented version number
<b>Public Patch</b>	An individual update to the product installation that is available to all Oracle Support accounts
<b>Patch Sets</b>	A collection of patches that can be installed as a unit and in a specific sequence
<b>Patch Profile</b>	A named grouping of related patches that have been applied to an installation
<b>Private Patch</b>	An individual update to the product installation that is granted only to specific Oracle Support accounts; requires a special ID and passcode



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

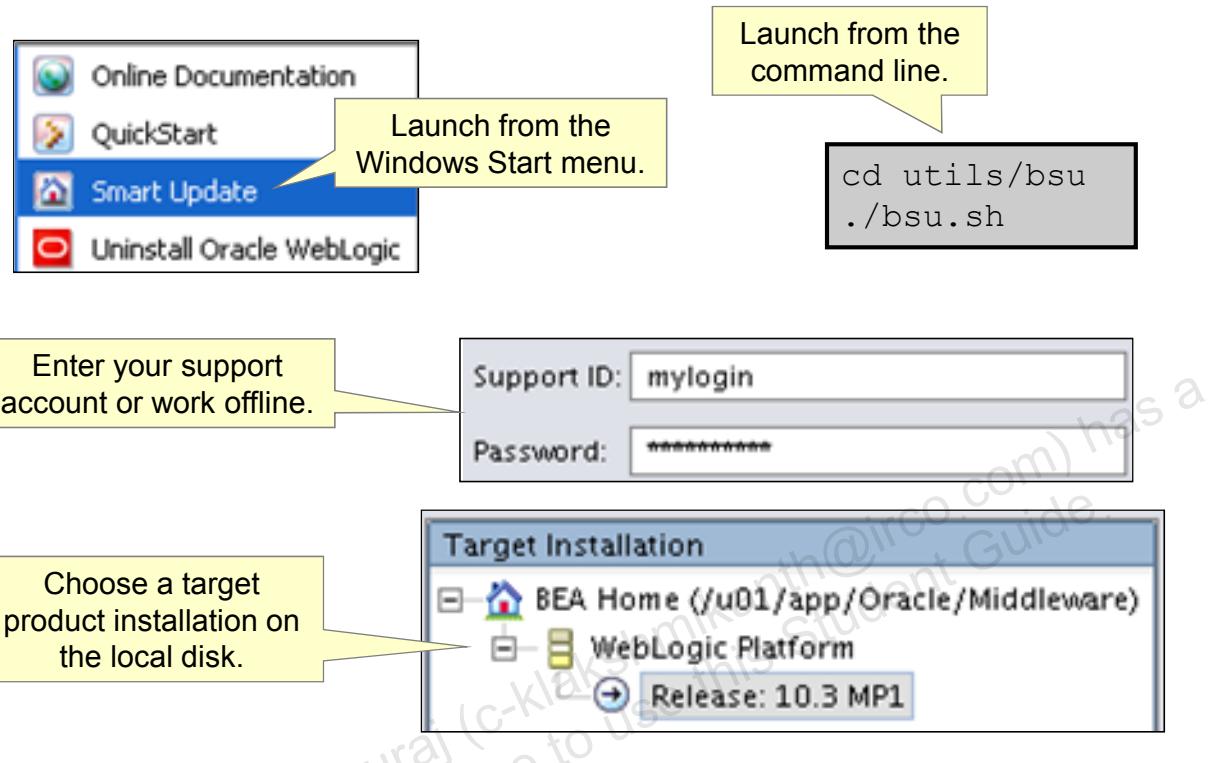
## WebLogic Patch Terminology

To download patches and maintenance packs using Smart Update, you need an Oracle Support login ID and password. Anyone with an account can obtain publicly available patches, but this account must also be linked to a valid support contract to obtain private patches and maintenance packs.

A maintenance pack is an update to an existing release that includes solutions to known problems and other product enhancements. A maintenance pack is not a replacement for an installation of the WebLogic product, but a package of changes and additions to it. Maintenance packs can also be downloaded manually from the Oracle Support site and installed without the use of Smart Update. Prior versions of WebLogic Server referred to these as “service packs.”

By default, when you download and apply patches, those patches are in effect for the entire target installation. Every domain and server that is configured to run from such an installation runs against the patches applied to it. Sometimes, however, individual servers or domains within a production environment need to run at different patch levels. To accommodate this need, Smart Update enables you to point individual domains, servers, or clusters at specific patches that are not necessarily in effect installation-wide. This is accomplished by defining custom patch profiles in Smart Update.

# Getting Started



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Getting Started

Smart Update is located in your Middleware Home directory, under the subdirectory `utils/bsu` (`bsu` = "BEA Smart Update"). On Windows environments, a Start menu shortcut to this file is also generated automatically.

You are automatically prompted for your support credentials when starting Smart Update. However, if you are not using Smart Update to download new patches or maintenance packs, you can instead opt to Work Offline in the login dialog. Remember that a support account is a prerequisite for downloading patches or for validating patches against the latest dependency and conflict data.

By default, the product installation used to launch the Smart Update tool is selected as the target installation. To specify a different product installation, you must open the Target Installation panel and highlight the appropriate entry in the list of product installations displayed there. Select File > Target Installation from the main menu. This list identifies the product installations detected on the current machine, including all Middleware Home directories.

If you are running Smart Update on a machine located behind a firewall, you may need to specify the location of your HTTP proxy server along with any credentials. From the main menu, select File > Preferences, and then click the Proxy tab.

# Downloading and Applying Patches

Patch ID	Description	Product	CR	Category	Select
Critical					
Optional					
327W	Advisory: Elevation o...	WebLogic ...	CR384662	Console	<input type="checkbox"/>
3QHE	Advisory: Security po...	WebLogic ...	CR380519	Web Services	<input checked="" type="checkbox"/>
51PX	Advisory: (Patch 1 of ...	WebLogic ...	CR382275	Portal Adminis...	<input checked="" type="checkbox"/>
5KXF	Patch to add Oracle ...	WebLogic ...	CR379710	General	<input type="checkbox"/>

Select patches to download.

Patch ID	Description	Product	CR	Category	Apply
3QHE	Advisory: Security poli...	WebLogic Ser...	CR380519	Web Services	
51PX	Advisory: (Patch 1 of ...	WebLogic Portal	CR382275	Portal Admi...	

Apply downloaded patch to target installation.

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

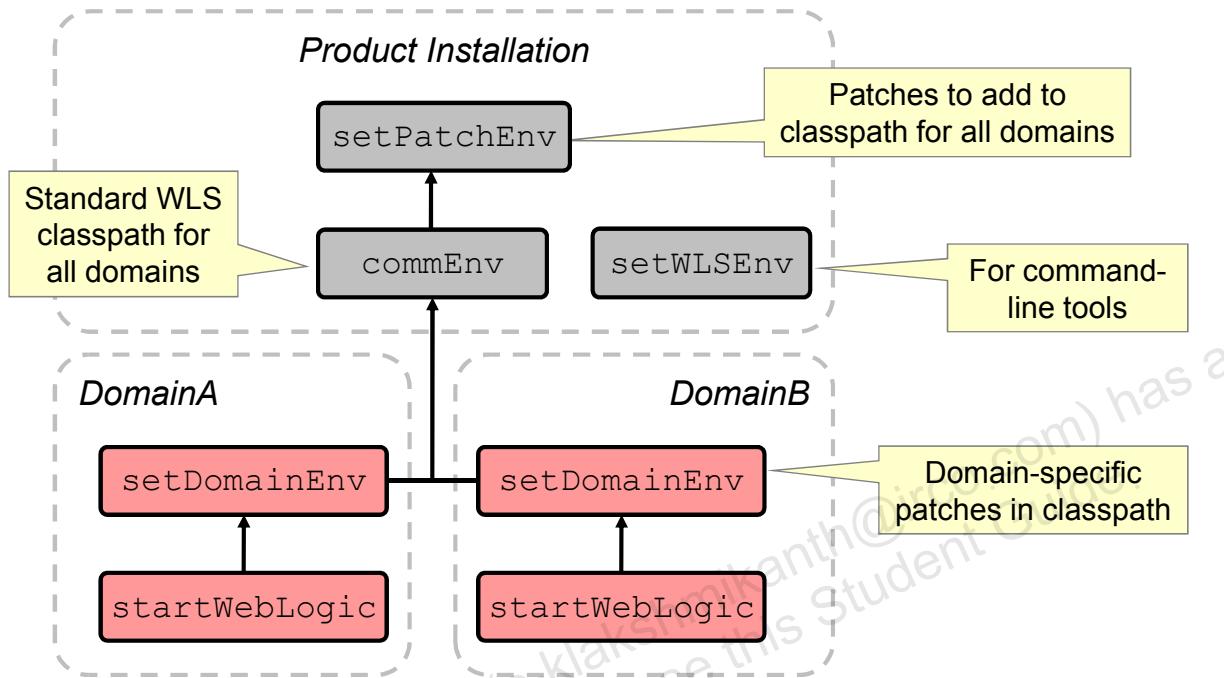
## Downloading and Applying Patches

To download one or more publicly available patches, complete the following procedure:

1. If not already done, log in to Smart Update and select a target installation in the left panel.
2. In the right panel, click the Get Patches tab.
3. In the Select column, use the provided check boxes to select the patches you want to download. You may click the Info icon next to a patch's ID to view detailed information about a specific patch.
4. Click the Download Selected button. You are prompted to indicate whether you want to check for conflicts before downloading the patches. If a conflict is detected (for example, the patch you are downloading requires that another patch be applied first), Smart Update reports the conflict. This gives you an opportunity to resolve the conflict before continuing, but does not prevent you from proceeding with the download.

To download a private patch, select Patches > Retrieve Private from the main menu. As always, to download any type of patch you must be first be logged into Smart Update using your support credentials. You will then be prompted to enter the ID and passcode of the desired private patch, which you must obtain from Oracle Support. After downloading, a private patch is validated and applied in the same fashion as public patches.

# WebLogic Start Script Review



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

ORACLE®

## WebLogic Start Script Review

To start a server within a domain, you can use the generated `startWebLogic` script or develop your own custom scripts. The default `startWebLogic` script executes your domain's `setDomainEnv` script. This script in turn calls a script named `commEnv`, which is included with your product installation. The `commEnv` script uses the `setPatchEnv` script, which is responsible for initializing variables that point to your currently installed patches. Another script that makes use of `commEnv` is `setWLSEnv`, which is not directly used to start servers. Instead, it provides a convenient way of initializing your environment to support WebLogic developer and administrator tools including Ant and WLST.

Some patches directly replace existing resources in your product installation and, therefore, become effective automatically as soon as they are applied. They are not enabled through a reference in a server start script, such as a classpath. Examples of patches that typically contain replacement artifacts include resources for Web server plug-ins, native socket multiplexers, and native dynamically linked libraries.

Other patches include Java class or library files that are loaded by server start scripts. These patch files are written by Smart Update to a special location within your Oracle root installation folder, <MIDDLEWARE\_HOME>. These folders typically start with the name "patch ".

## Patch Classpath

By default, Smart Update:

- Adds JAR files for applied patches to the server classpath for all domains
- Updates the MANIFEST within the `weblogic_patch.jar` file



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Patch Classpath

The `commEnv` script calls the `setPatchEnv` script, which includes default definitions for the environment variables that specify the locations of patch JAR files. By default, these patch variables are in effect for every WebLogic Server instance that is started using `commEnv`.

The `setPatchEnv` script variable `PATCH_CLASSPATH` defines a single JAR file named `weblogic_patch.jar`. This file contains no Java classes, but includes a manifest file. The manifest file lists all the specific patch JAR files that are to be dynamically included in the server classpath. As new patches are downloaded and applied using Smart Update, this manifest file is updated. With this approach, scripts such as `commEnv`, `setDomainEnv`, and `startWebLogic` need not be modified.

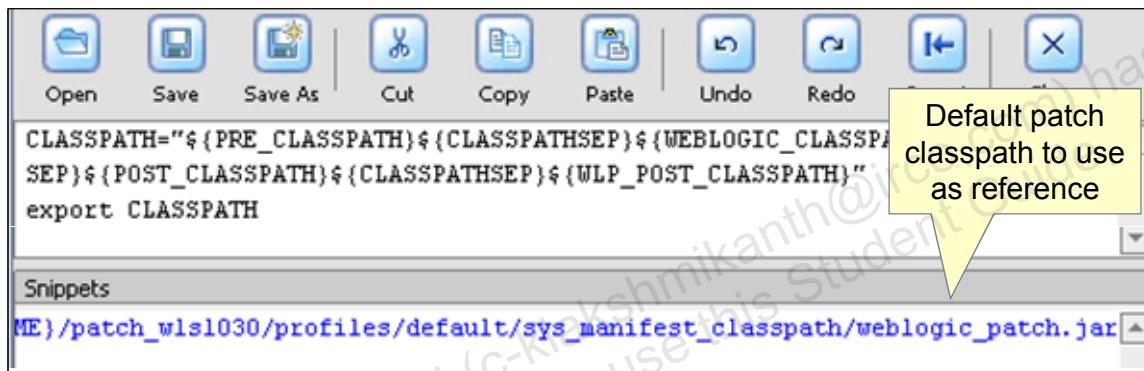
Alternatively, you can customize a domain's `setDomainEnv` script to contain its own definition for the `PATCH_CLASSPATH` variable. In this scenario, the definition of `PATCH_CLASSPATH` in the `commEnv` script is overridden for those server instances. It is important that if you add a patch path variable definition to a start script, the definition is placed before the statement that invokes another start script.

## Start Script Editor

You may need to manually edit start scripts to apply a patch if:

- You want to limit which domains use a patch
- You have custom scripts that do not use `commEnv`

Smart Update includes a script editor to help with this task.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Start Script Editor

The Start Script Editor is a tool for locating start scripts in your environment and for creating definitions for patch path variables in them. Smart Update maintains scripts in multiproduct environments with different patch levels as expressed in combinations of default patch profile and custom patch profiles.

1. Select a target installation from the Target Installation panel.
2. Select Patches > Start Script Editor from the main menu.
3. Choose the Patch Profile to which the domain or server will point, if multiple profiles are defined.
4. Choose the Product for which you are editing a start script, if multiple products are installed.
5. Click Open and select the start script that you want to modify.
6. Add the appropriate patch path variable definition to your patch profile. The editor provides code snippets with suggested definitions for the `PATCH_CLASSPATH` and other variables, all of which are customized for the previously selected patch profile. You may want to modify these definitions, however, depending on your needs.

## Smart Update Command-Line Interface

- The Smart Update command-line interface:
  - Can view, add, remove, and distribute patches
  - Cannot be used to download new patches or create custom patch profiles
- Smart Update command-line examples:

```
bsu.sh -install -patchlist=3QHE,51PX  
        -prod_dir=/home/oracle/wlserver_10.3 -verbose  
  
bsu.sh -remove -patchlist=5KXF  
        -prod_dir=/home/oracle/wlserver_10.3 -verbose  
  
bu.sh -view -status=downloaded  
        -prod_dir=/home/oracle/wlserver_10.3  
  
bsu.sh -report
```



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Smart Update Command-Line Interface

The Smart Update application `bsu.cmd/sh` can also apply patches that have been downloaded into a patch download directory interactively from the command line or via a script. This tool allows you to create an automated mechanism for replicating a specific maintenance level of a product that is installed on multiple machines. This capability is especially valuable in production environments, in which the distribution of software updates to machines must be implemented in a controlled, reliable, and reproducible manner.

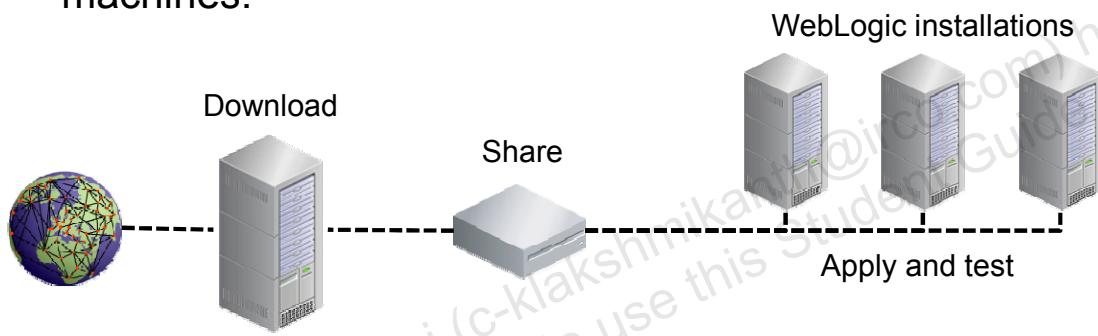
The `-patch_download_dir` argument specifies the directory from which the patches are to be applied. If you do not specify this argument, the patch download directory designated in the preference dialog box of the interface is used instead. However, if no preference is set, the download directory is `utils\bsu\cache_dir`.

The `-profile` argument specifies a custom patch profile name. The default profile is used if not specified.

The `-log` argument specifies the name and location of a log file to generate. The `-log_priority` argument specifies the priority of log information to be captured ("debug," "info," "warn," and so on). The default priority is "debug."

# Patch Distribution

1. Use a dedicated, secure machine to download patches.
2. Map the patch download directory to shared network storage.
3. Apply patches on each machine from the shared download directory using a script.
4. Test thoroughly before running the script on production machines.



ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Patch Distribution

In many enterprises, updates obtained from a vendor are tightly controlled and managed. For example, they might be kept in a secure repository that few individuals in MIS are authorized to access. Copying or downloading updates into the repository might be subject to rigorous approval, scheduling, auditing, and logging procedures. Machines in the production environment may not obtain updates from the Internet. In fact, the machines may not be connected to the Internet at all. Instead, updates are made available only from a specific location, or set of locations, in the enterprise. And the downloading of updates to the production machines are regulated by several business practices and procedures.

Similarly, updates are promoted to production systems in specific stages during which they are thoroughly tested. For example, an update might first be installed on a single machine, separated from the production environment, in which production applications are tested to ensure that the update works as expected and does not introduce regressions. Before being rolled out into the production environment, the updates may be tested in an intermediate staging area, where they are subjected to loads that mimic expected usage in the production environment.

Remember that you can use the File > Preferences menu or the `-patch_download_dir` command-line argument to specify a shared network location in which to store and retrieve patches managed by Smart Update.

# Oracle Cloning Tool

- Oracle products also include cloning tools to:
  - Archive your entire product installation, excluding domains, logs, and other temporary files
  - Extract the archive on another machine
- Source and destination machines must use the same installation paths.
- Cloning a WebLogic product installation:

```
cd utils/clone  
clone.sh ${JAVA_HOME} my_wls_install.jar  
  
restore.sh ${JAVA_HOME} my_wls_install.jar
```



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Oracle Cloning Tool

For WebLogic Server, the cloning tool does not tokenize any scripts found in your product installation. Therefore, a cloned copy of your installation will not function correctly if placed at a different location on the file system than that of the original source installation. For other Oracle products, however, the cloning tool may support variable replacement for installation files.

You can customize the list of folders and files that the cloning tool will include in the archive using the `utils/clone.xml` configuration file. You may need to edit this file if, for example, your product installation contains a new root product patch directory.

# Quiz

Name three capabilities of Smart Update.

- a. Update a domain's configuration.
- b. Download a maintenance pack.
- c. Apply a product patch.
- d. Add a patch to the server classpath.
- e. Create a domain from a template.



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

**Answer:** b, c, d

# Quiz

What file is responsible for including patch JAR files in the server classpath?

- a. nodemanager.properties
- b. weblogic\_patch.jar
- c. maintenance.jar
- d. setWLSEnv.sh
- e. boot.properties

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

**Answer: b**

## Summary

In this lesson, you should have learned how to:

- List the capabilities of Smart Update
- Use the Smart Update graphical or command-line interfaces
- Download and apply patches
- Explain how patches are linked to server start scripts
- Describe the use of the cloning tool

## Domain Templates

ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

# Objectives

After completing this lesson, you will be able to:

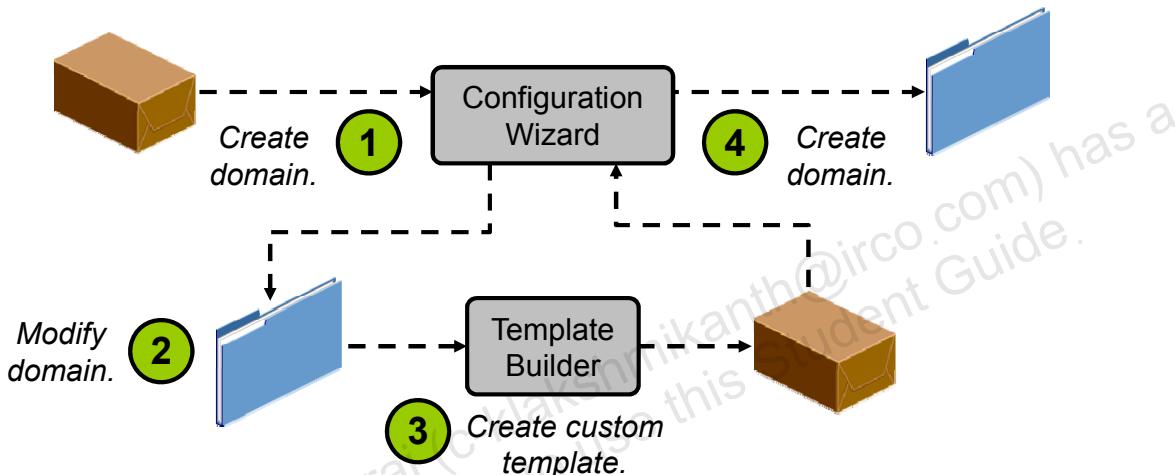
- List the components of a typical domain template
- Configure variable replacement in template files
- Create domain and extension templates using Template Builder

# Road Map

- Template Concepts
  - Template Contents
  - Replacement Variables
  - SQL Scripts
  - Template Types
- Template Builder

## Domain Template Review

- The Fusion Middleware Configuration Wizard creates WebLogic domains based on templates.
- Using custom templates, you can distribute a baseline domain configuration across multiple projects.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Domain Template Review

A domain is the basic administration unit of WebLogic Server. It consists of one or more WebLogic Server instances, and logically related resources and services that are managed, collectively, as one unit. In addition to infrastructure components such as servers and clusters, a domain defines application deployments, supported application services (such as database and messaging services), security options, and physical host machines.

The Configuration Wizard guides you through the process of creating a domain for your target environment by selecting the product components that you want to include in your domain, or by using domain templates. If required, you can also customize the domain to suit your environment by adding and configuring managed servers, clusters, and machine definitions, or customizing predefined JDBC data sources, and JMS file store directories.

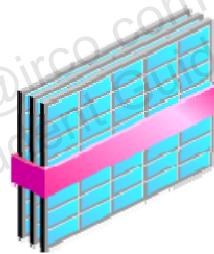
A domain template defines the full set of resources within a domain, including infrastructure components, applications, services, security options, and general environment and operating system options. You can create this type of template from an existing domain by using the Domain Template Builder, WLST, or the pack command-line tool. Subsequently, you can create a domain based on the template by using the Configuration Wizard or WLST.

The product installation includes a set of predefined domain and extension templates. This set includes the base WebLogic Server domain template and various extension templates that allow you to add component features and samples to the base domain.

## Template Contents

A domain template contains some combination of the following:

- Domain configuration (`config.xml`) and supporting resource modules (JDBC, JMS, diagnostics)
- Java EE applications and shared libraries
- Initial users, groups, and roles to add to the security realm
- Windows and UNIX server start scripts
- SQL scripts to initialize supporting databases
- Windows Start menu entries
- Other custom folders and files



ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Template Contents

To create a new domain template, perform the following steps:

1. Select the domain template or the directory of the domain from which you want to create a new domain template.
2. Specify a description of the template.
3. Review and modify, if required, the list of applications to be included in the template. For example, you can include a WLST script named `final.py` to execute after the domain is created.
4. Review and modify, if required, the files to be included in the template.
5. Add SQL scripts for each database that you expect to be used with the domains created from this template and specify the order in which the scripts are executed. Your domain must contain at least one data source to use this capability.
6. Define parameters for the administration server.
7. Specify a username and password to be used for starting the administration server. Optionally, you can also configure additional security features by defining users and groups and assigning them to global security roles.
8. Optionally, define entries for the Windows Start menu.

## Template Exclusions

Generated domain templates automatically exclude the following domain files:

- The contents of the domain's `servers` folder (logs, cached application resources, cached LDAP data, and so on)
- Locks and other temporary files



ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Script Replacement Variables

- All literal paths in standard server start scripts are automatically replaced with variables by the Template Builder.
- The Configuration Wizard replaces these variables with paths that correspond to the local product and domain installation.

```
WL_HOME="/u01/oracle/product/fmw/wlserver_10.3"
```



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

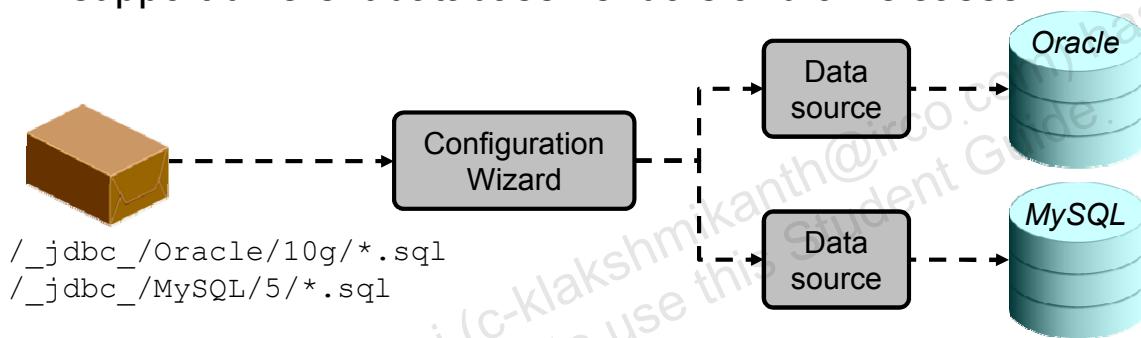
### Script Replacement Variables

When you are creating a template, you want the scripts and files that you are packaging with your template to be free of local domain environment settings and ready for use by the Configuration Wizard. The Domain Template Builder automatically updates any standard scripts included in a template, such as start scripts, by replacing hard-coded values for various domain environment settings with replacement variables. The Configuration Wizard can later replace these variables with new hard-coded values during the configuration of a new domain.

Custom scripts and other files are not automatically updated with replacement variables. The Domain Template Builder interface includes a script editor to help with this task. Select the file to edit, highlight the literal string to replace, and select from a list of defined variables to replace the string with. Commonly used variables include `DOMAIN_NAME`, `DOMAIN_HOME`, `JAVA_HOME`, and `WL_HOME`. The list of files for the Configuration Wizard to update along with the variables to replace is stored within the template as a file named `stringsubs.xml`.

## Template SQL Scripts

- Templates that include at least one data source can also contain SQL scripts to initialize database resources.
- The Configuration Wizard will use the connection parameters of the selected data source to execute the scripts.
- A single template can include multiple script versions to support different database vendors and/or releases.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Template SQL Scripts

The Add SQL Scripts window within the Template Builder prompts you to add SQL scripts for each database that you expect to be used with the domains created from this template. You can also specify the order in which the scripts are executed. When you create a domain based on this template, the databases and associated SQL scripts that you include are displayed in the Run Database Scripts window of the Configuration Wizard.

Perform the following steps in the Template Builder:

1. In the left pane of the Add SQL Scripts window, select a database from the Type drop-down list.
2. Select the database version from the Version drop-down list, or enter a version number directly in the field.
3. Click Add SQL File. Navigate to the directory that contains the SQL scripts for the selected database. Select the SQL files to be added, and click Add SQL File(s).
4. The right pane displays a tree view of all the databases and associated SQL scripts included in the template. You can change the order in which the SQL files are executed by using the Up and Down arrow icons.
5. Repeat these steps for each database for which you want to include SQL files. You may also remove existing SQL scripts from the template if you want.

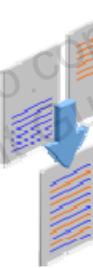
## Template SQL Scripts (continued)

Alternatively, you may want to run SQL scripts not present within the template JAR itself and instead found at some other location on the file system. Create a file named `jdbc.index` and within it list the names and locations of the SQL files to run on the local file system. Use a tilde (~) as a shortcut for your `MIDDLEWARE_HOME` directory. You can similarly use any available replacement variables within `jdbc.index`, but be sure to also register the file for variable replacement in the Template Builder. Similar to template SQL scripts, these index files are placed within a folder that corresponds to the database vendor and version that you selected while creating the template (for example, “`_jdbc_Oracle/10g`”).

## Extension Template Concepts

An extension template:

- Can be created from an existing domain or from another domain template
- Includes a `config.xml` whose contents are merged with the target domain's configuration
- Does not include definitions for servers, clusters, or machines (removed by the Template Builder)



ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Extension Template Concepts

Extension templates define applications and services that can provide additional features for a domain, such as JDBC or JMS components. This type of template can be used to update an existing domain. The product installation includes a set of predefined domain and extension templates. This set includes the base WebLogic Server domain template and various extension templates that allow you to add component features and samples to the base domain. But you may also use the Template Builder tool to create your own custom extension templates as well.

# Fusion Middleware (FMW) Templates

- A simple base WebLogic Server domain template is bundled with the product at <WL\_HOME>/common/templates/domains.
- Other FMW products install additional extension templates:
  - To initialize domains with product-specific resources and applications
  - At <PRODUCT\_HOME>/common/templates/applications



ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Fusion Middleware (FMW) Templates

When additional Oracle FMW products have been installed, the Select Domain Source window in the Configuration Wizard will include check boxes that correspond to the components of these products. Each component is typically associated with a domain or an extension template.

When creating your domain, if you choose not to configure all the components in a given product suite, such as Oracle SOA Suite, you can add these components at a later date by extending your domain.

One of these product components is Java Required Files (JRF), which consists of those resources not included in the default Oracle WebLogic Server installation and which provides common functionality for other FMW products and application frameworks.

## Section Summary

In this section, you should have learned how to:

- Describe the contents of a typical domain, an extension, or a managed server template
- Explain how start scripts and SQL scripts are handled by the Template Builder and Configuration Wizard

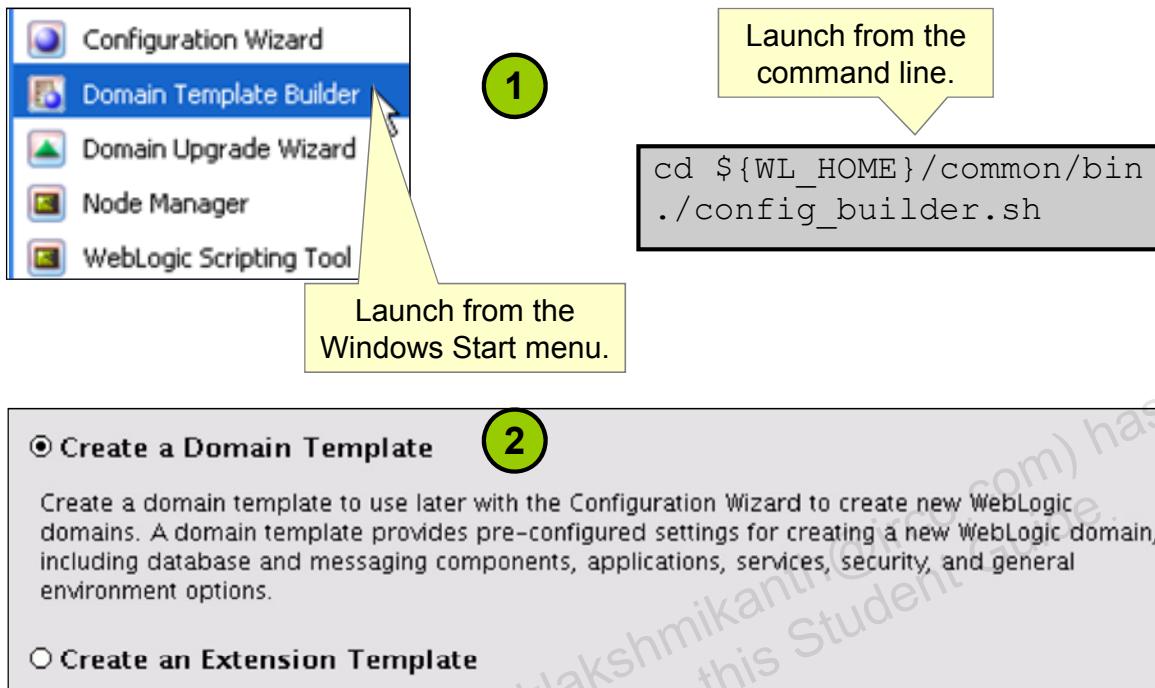
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

ORACLE®

# Road Map

- Template Concepts
- Template Builder
  - Select Template Source
  - Add Custom Files
  - Add SQL Scripts
  - Edit Security Entities
  - Configure Replacement Variables

# Creating a Domain Template



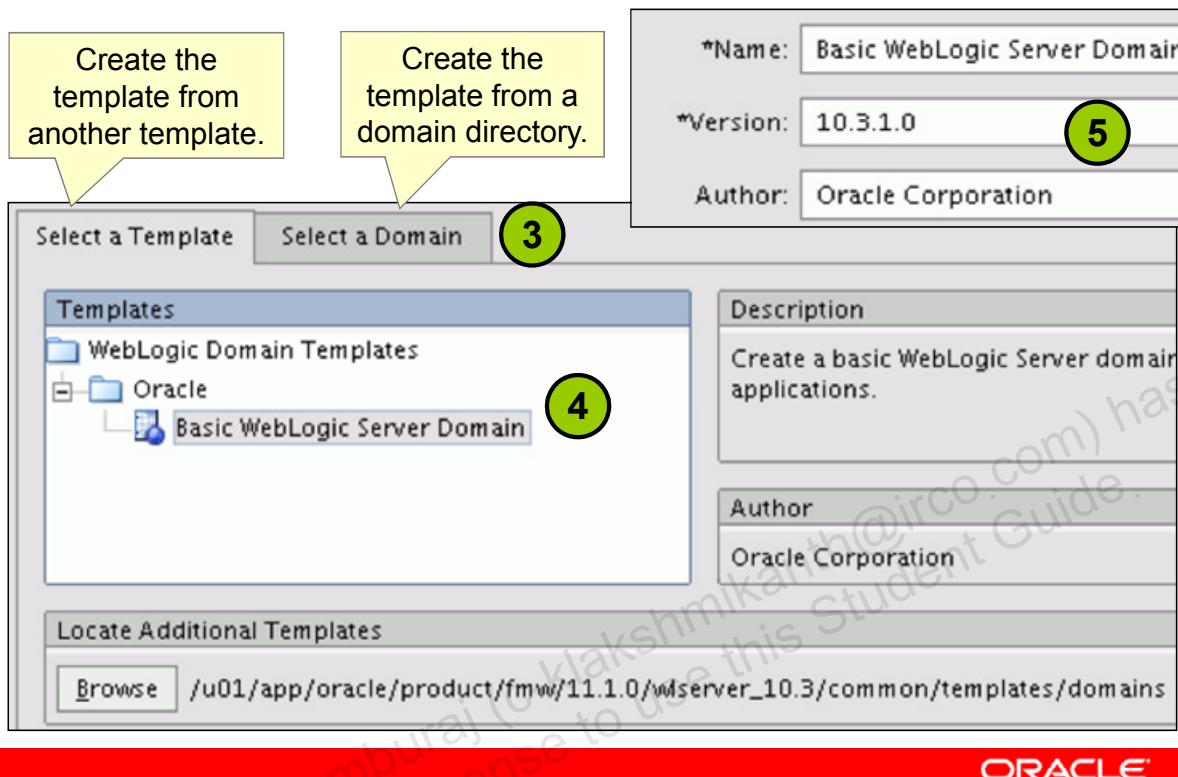
ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Creating a Domain Template

1. Launch the Domain Template Builder tool.
2. To create a full domain template, select the “Create a Domain Template” option and click Next.

## Selecting the Template Domain Source

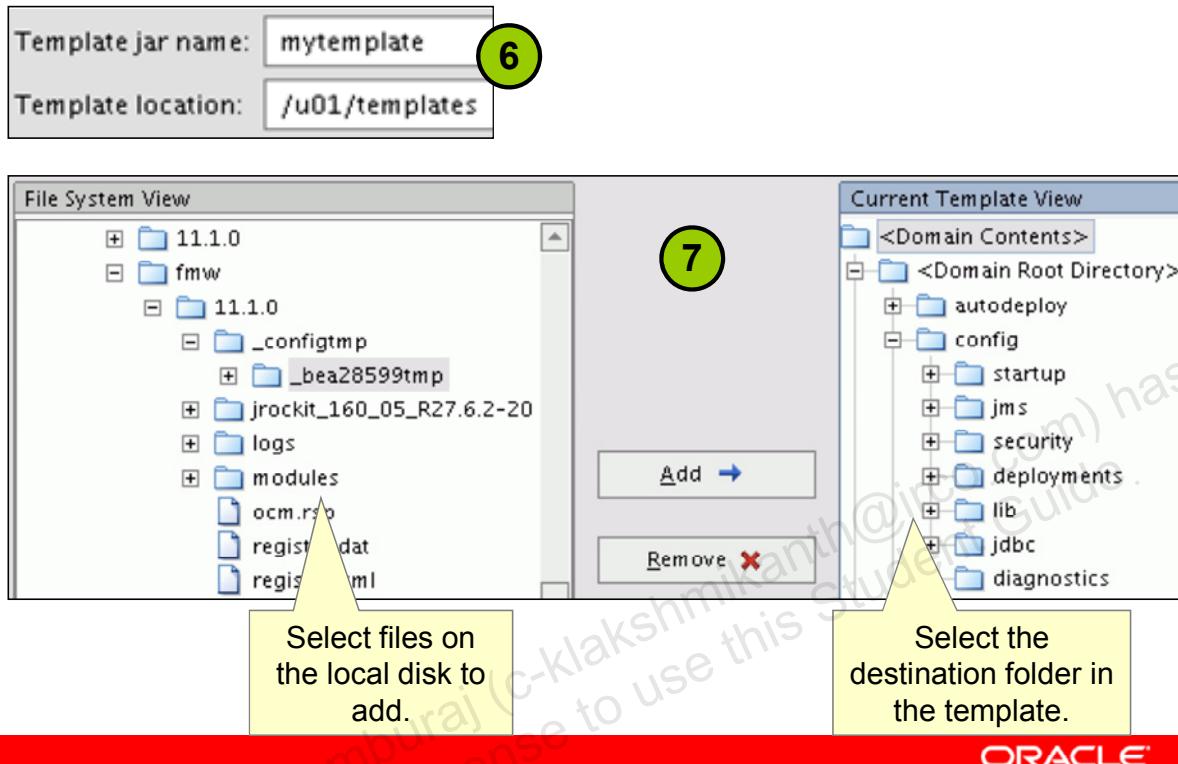


Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Selecting the Template Domain Source

3. Choose from the following tabs: “Select a Template” and “Select a Domain.”
4. Select the domain or domain template from which you want to create a domain template. The templates to choose from are located in the directory specified in the Locate Additional Templates pane. If you want to change the directory, click Browse, and then navigate to the appropriate directory or enter the path manually. When finished, click Next.
5. Enter a name and descriptive metadata about the new template. If you selected a template as the source for the new template, information about the selected template is displayed. Review the information, and, if necessary, change it to suit the requirements of your domain. Then click Next.

## Selecting the Template Destination: Adding Files to the Template



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

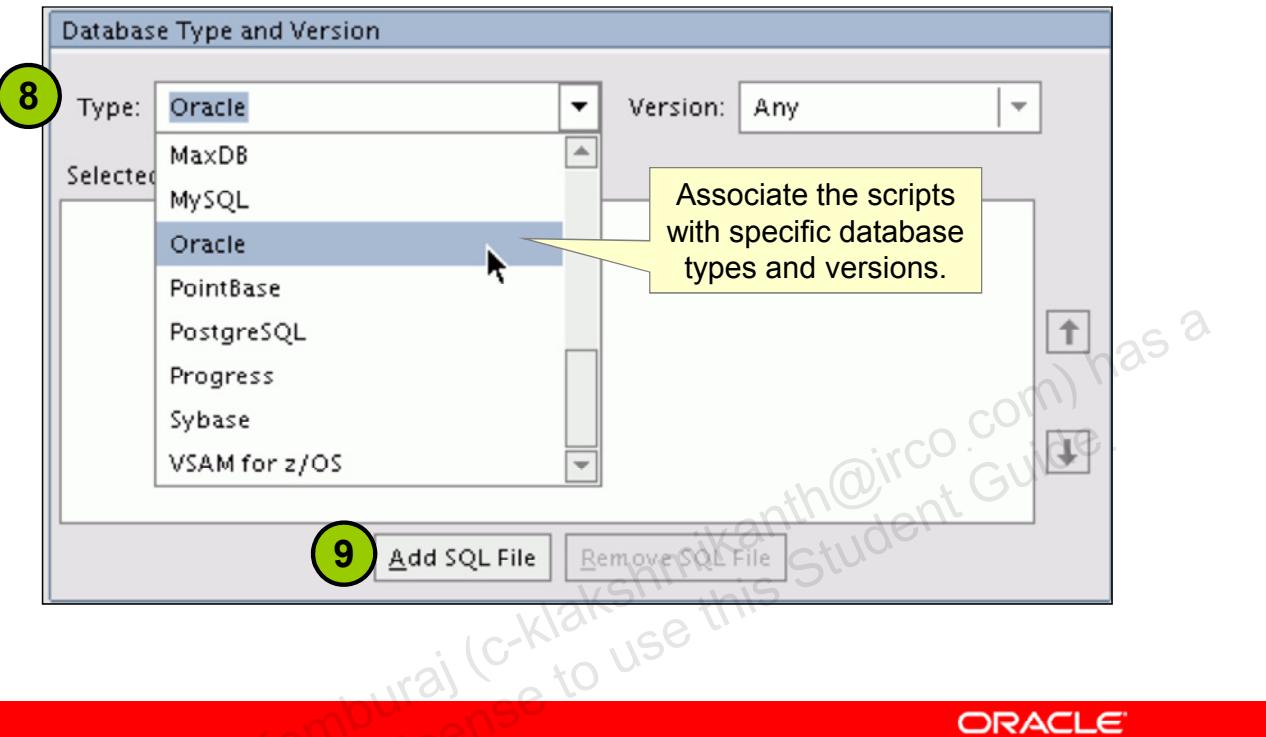
## Selecting the Template Destination: Adding Files to the Template

6. Enter the Template jar name and Template location. Click Next.
7. The Add Files page enables you to review, add, or remove files in the template. Select a file or folder in the File System View pane, select the destination folder in the Current Template View pane, and then click the Add button. When finished, click Next.

By default, the Domain Template Builder includes files from the domain or template that you specified as the source for the new template:

- If you selected an existing template as the source for the new template, all the files from the source template are automatically included. If the existing template defines a separate applications directory, the applications in the template are listed under the Applications Root Directory in the Current Template View pane.
- If you selected a domain as the source for your new template, the following files and directories are included by default for your domain:
  - All the files in the root directory with the following extensions: .cmd, .sh, .xml, .properties, and .ini
  - Any file with the extension .pem defined in the secure sockets layer (SSL) configuration
  - /bin and /lib directories
  - All the files in /security that are not created automatically during domain creation

## Adding SQL Scripts to the Template



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Adding SQL Scripts to the Template

8. Select a database from the Type drop-down list. Then select the database version from the Version drop-down list or enter a version number directly in the field.
9. Click Add SQL File. Navigate to the directory that contains the SQL scripts for the selected database. Select the SQL files to be added, and click Add SQL File(s). Repeat the previous steps to add scripts for additional database types and/or versions. You can also change the order in which the SQL files are executed by using the up and down arrows. The specified sequence is reflected in the Selected Database Scripts pane. When finished, click Next.

# Configuring the Administration Server

*Name:	MedRecAdmSvr
*Listen address:	All Local Addresses
Listen port:	7020
SSL listen port:	N/A
SSL enabled:	<input type="checkbox"/>

Change the current server settings, if desired.

*User name:	weblogic
*User password:	*****
*Confirm user password:	*****
Description:	This user is the default administrator.

Configure additional users, groups, and global roles.

No     Yes

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Configuring the Administration Server

10. Optionally, update the current administration server address and port(s). Click Next.
11. Enter the domain's administration credentials if not supplied in the source template. If the credentials are already set, you can optionally modify them.
12. If you want to add, edit, or remove additional users, groups, and/or roles, select the Yes option. Click Next.

When you create a domain template, the administrator username and password from the original domain or template are included in your new template. You can modify this username and password if required. In addition, you can provide extra security for application resources by using the following security features:

- Users and groups: Classification of individuals and groups of individuals who may be granted a security role. Typically, a group is a collection of users who share a role or function within a company, such as working in the same department.
- Global security roles: Dynamically computed classifications of users. Privileges are granted to or withheld from users according to the roles that they are assigned.

## Defining Users, Groups, and Roles

Add/remove/edit users or the names of groups and roles.

Name*	Description
1 Administrators	Administrators can view and m...
2 Deployers	Deployers can view all resourc...
3 Operators	Operators can view and modify...
4 Monitors	Monitors can view and modify...
5 AppTesters	AppTesters group.
6 CrossDomainConnectors	CrossDomainConnectors can ...
7 AdminChannelUsers	AdminChannelUsers can access ...
8 OracleSystemGroup	Oracle application software sy...

ORACLE®

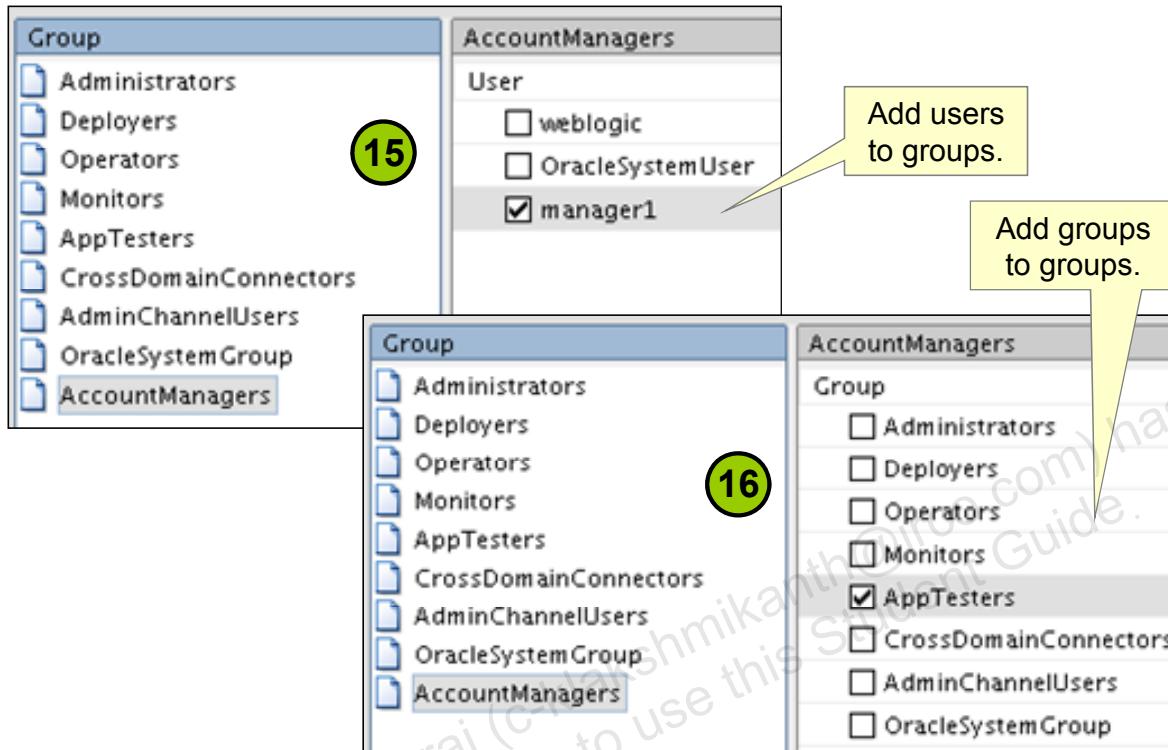
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Configuring Users, Groups, and Roles

13. Click the User tab and review the current user configuration. Click the Group tab and review the current group configuration. Click the Role tab and review the current role configuration.
14. For each tab, edit the existing entities, click Add to create a new entity, or click Delete to remove an existing entity. For users, enter the username and password. For groups and roles, simply enter the name of the group or role. When finished, click Next.

Depending on the source template or domain selected, one or more users, groups, or roles may be already defined. In addition, Oracle WebLogic Server defines a default set of groups and roles.

# Configuring Group Membership



**ORACLE**

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Configuring Group Membership

15. Optionally, select a group and update the list of users that are members of the group. When finished, click Next.
16. Optionally, select a group in the left pane and update the list of other subgroups that are members of it. For example, suppose the group NY is a member of the group USA. Users then can be assigned to states such as NY and implicitly also belong to countries. When finished, click Next.

Domains, by default, always include groups related to administrative tools access. For example, the following groups are always present:

- Administrators
- Operators
- Deployers
- Monitors
- AppTesters
- Everyone (implicit and, therefore, not listed)

# Configuring Role Membership

The screenshot shows the 'Role Membership' configuration for the 'AppManager' role. On the left, a list of roles is shown, with 'AppManager' selected. A green circle with the number '17' is overlaid on this list. On the right, the 'User & Group' configuration pane is displayed. Under the 'Group' section, the 'AccountManagers' checkbox is checked. A yellow callout bubble with the text 'Add groups or users to roles.' points to this checkbox. Other options listed include 'Administrators', 'Deployers', 'Operators', 'Monitors', 'AppTesters', 'CrossDomainConnectors', 'AdminChannelUsers', 'OracleSystem Group', 'everyone', 'users', and 'User'. The 'weblogic' option is also present under 'User'.

ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

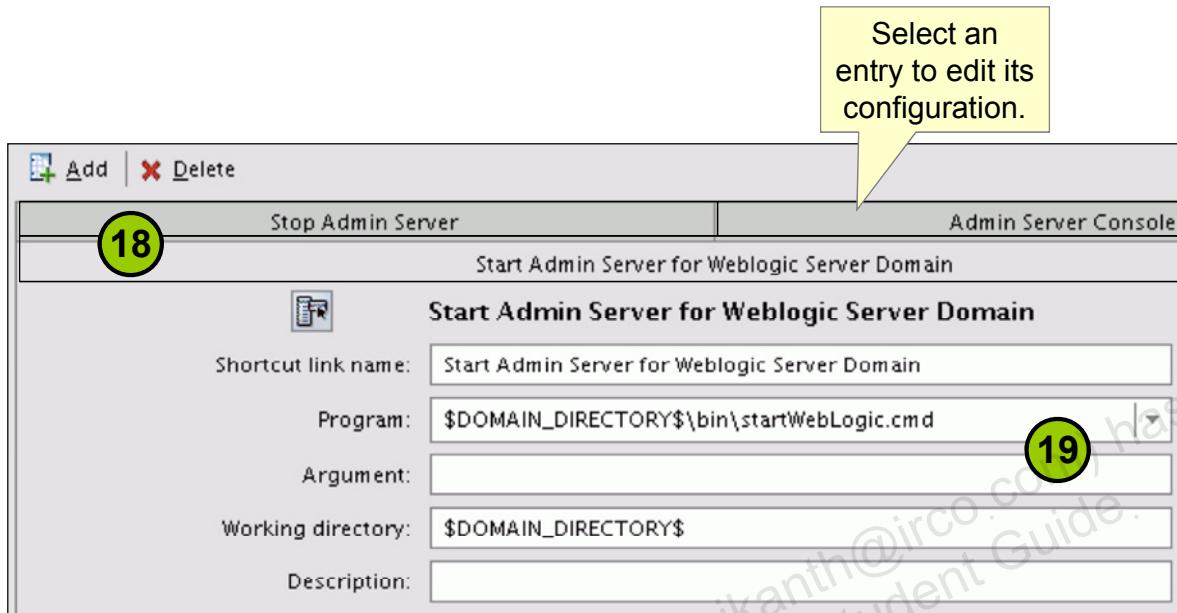
## Configuring Role Membership

17. Optionally, select a role in the left pane and update the list groups and/or users that are members of it. The Template Builder does not support other types of role expressions, such as date and time. When finished, click Next.

Similar to groups, default roles related to administrative tools' access permissions are always present. Similarly, another role called "Anonymous" is always present and is used to create policies for all unauthenticated users.

Do not make the default global security roles for administrative and server resources more restrictive. If you eliminate any existing security roles, you risk degrading the Oracle WebLogic Server operation. You can, however, make the default security roles more inclusive (for example, by adding new security roles).

## Creating Windows Start Menu Entries



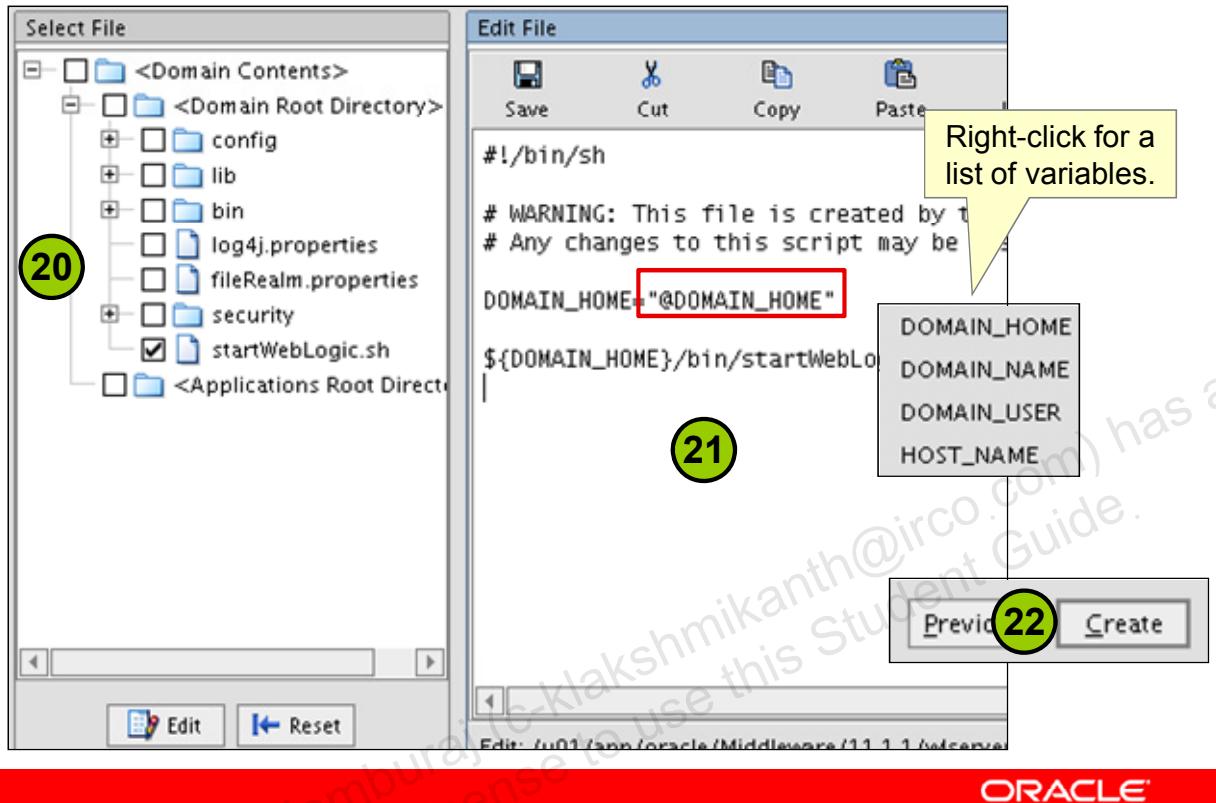
ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Creating Windows Start Menu Entries

18. Click Add to create a new Windows Start menu entry for this domain template. Multiple Start menu entries are shown as tabs at the top of the page. If you selected a template as the source for your custom template, any existing entries from the selected template are displayed. Review these entries and modify them if necessary.
19. After clicking a tab, enter the details of the shortcut to be placed in the Start menu. Use script replacement variables as necessary. These variables will be later replaced by the Configuration Wizard, similar to other server start scripts. When finished, click Next.

## Preparing Scripts and Files with Variables



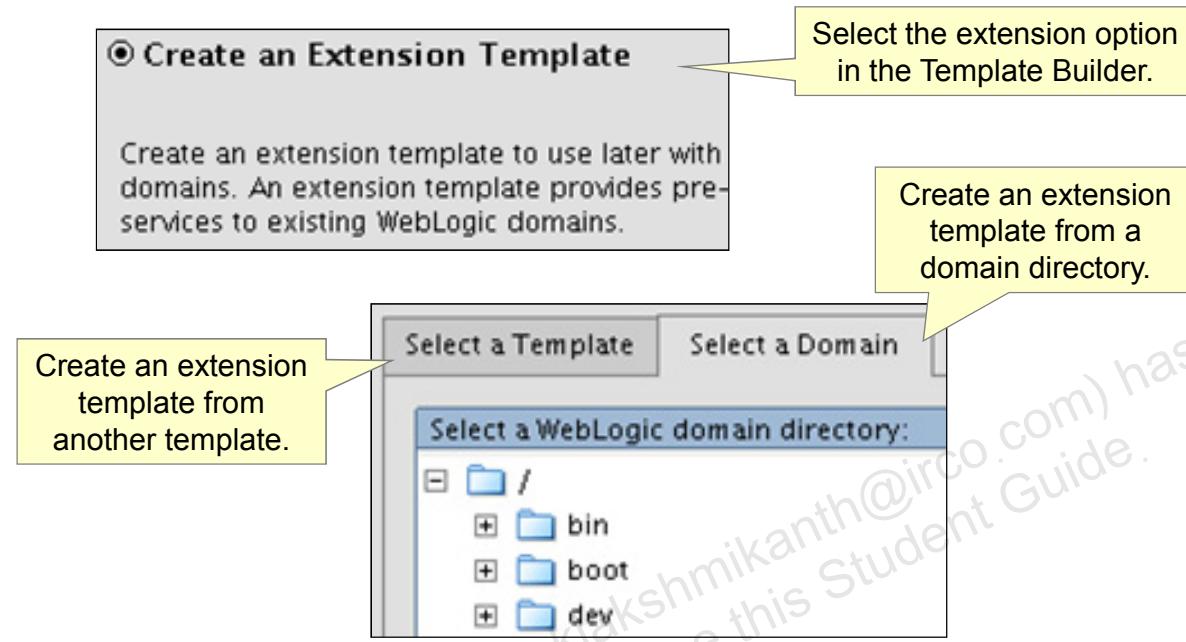
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

ORACLE

### Preparing Scripts and Files with Variables

20. Use the check boxes in the left pane to select the templates file that require variable replacement by the Configuration Wizard. The check boxes for files that were automatically updated by the Domain Template Builder will be already selected. To update a file and insert replacement variables within it, either double-click the file, or select the file and click Edit.
21. In the editor pane on the right side, insert variables as required. For convenience, right-click and select from the list of common replacement variables. After you finish editing the file, click Save. After all the editing is complete, click Next.
22. Review the configuration summary of your new template and click Create.

## Creating an Extension Template



ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

# Creating an Extension Template

1. Select the template source.
2. Select the applications in source to keep/omit.
3. Add any custom files.
4. Add any SQL scripts.
5. Optionally, add or edit security users, groups, and/or roles.
6. Add replacement variables to any custom files.



ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Creating an Extension Template

1. The Select a Template Domain Source window prompts you to select the application template or domain directory from which you want to create an extension template. The Describe the Template window then prompts you to provide a description for the template.
2. The Add or Omit Applications window allows you to change the relative directory into which applications are imported into template JAR. Or you may remove existing applications from the template.
3. By default, the Domain Template Builder includes files from the domain or extension template that you specified as the source for the new extension template. In the Add Files window, you can review, add, and remove files in the template.
4. In the Add SQL Scripts window, you can add SQL scripts for each database that you expect to be used with the domains that you extend using this template.
5. In the Security Configuration Options window, you can set security options for your application. If you choose No, the security settings for the new extension template are the same as the settings defined in the source template or domain.
6. The Prepare Scripts and Files with Replacement Variables window allows you to replace hard-coded strings with replacement variables in files that have not been automatically updated by the Domain Template Builder.

## Section Summary

In this section, you should have learned how to:

- Use the Template Builder to assemble a domain or extension template
- Add server scripts, SQL scripts, and other custom files to a template
- Add user, group, and role definitions to a template
- Edit a template file to use replacement variables



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

# Quiz

Which of the following is NOT a feature of domain templates?

- a. SQL scripts
- b. Start menu entries
- c. Email notifications
- d. Variable replacement



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

**Answer: c**

## Quiz

Which two types of domain resources are NOT included in an extension template?

- a. Data Source
- b. Cluster
- c. Application
- d. JMS Module
- e. Server



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

**Answer: b, e**

## Lesson Summary

In this lesson, you should have learned how to:

- Explain the use of domain templates in your enterprise
- Build and use custom domain and extension templates
- Initialize a database via a domain template



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Practice 3-1: Create a Custom Domain Template

This practice covers the following topics:

- Using the Domain Template Builder to create a template from an existing domain
- Adding custom files and SQL scripts to a template
- Configuring variable replacement in template files
- Creating a new domain from a custom template
- Using the patient features of the Medical Records (MedRec) application



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Other Domain Tools



ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

# Objectives

After completing this lesson, you will be able to:

- Describe some best practices for writing WLST scripts
- Create and apply templates using WLST
- Migrate domains using the pack/unpack tools
- Periodically back up a domain's configuration
- Track the changes made to a domain

# Review of Basic WLST Commands

Command	Description
<code>connect</code>	Connect to a server by using supplied credentials.
<code>disconnect</code>	Disconnect from the current server.
<code>getMBean</code>	Retrieve the MBean object at the given location.
<code>cd</code>	Navigate to the MBean at the given location and update the current management object ( <code>cmo</code> ) global variable.
<code>ls</code>	Print the attributes and values of the <code>cmo</code> MBean.
<code>pwd</code>	Print the current location in the MBean hierarchy.
<code>startEdit</code>	Begin a new change session.
<code>activate</code>	Commit all changes in the current session.
<code>deploy</code>	(Re)Deploy an application to servers.
<code>undeploy</code>	Shut down a running application on servers.



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Review of Basic WLST Commands

WLST enables you to navigate a hierarchy of MBeans in a similar fashion to navigating a hierarchy of files in a file system. For example, to navigate back to a parent MBean, enter the `cd('..')` command.

In the WLST file system, MBean hierarchies correspond to drives, MBean types and instances are directories, and MBean attributes and operations are files. In WLST, you traverse the hierarchical structure of MBeans by using commands such as `cd`, `ls`, and `pwd` in a way similar to how you would navigate a file system in a UNIX or Windows command shell.

After navigating to an MBean instance, you can interact with the MBean by using other WLST commands, such as `get`, `set`, and `invoke`. The built-in variable `cmo` is set to the current MBean each time you change directory. With this variable, you can perform operations directly against the current management object. However, an alternative approach is the `getMBean` command, which returns the management object for a given WLST path. Unlike `cd`, `getMBean` does not change the current path pointed to by `cmo`.

## WLST Best Practice: Variable Declaration

- Use variables in scripts, instead of hard-coding MBean attribute values.
- Declare variables at the top of scripts so that they can be easily modified.

```
url = 'localhost:7020'  
username = 'user'  
password = 'password'  
dsName = 'MyDS'  
targetName = 'MyServer'  
  
connect(username, password, url)  
...  
jdbcSystemResource = create(dsName, 'JDBCSystemResource')  
targetServer = getMBean('/Servers/' + targetName)  
jdbcSystemResource.addTarget(targetServer)
```



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### WLST Best Practice: Variable Declaration

Parameterize your WLST scripts. Provide variables defined and initialized in a preamble at the top of the script. This practice promotes reuse and can aid in troubleshooting.

Alternatively, you can define variables in a separate text file and import them as variables by using the `loadProperties` WLST command. For example:

```
loadProperties('c:/temp/myLoad.properties')
```

# WLST Best Practice: Password Management

To avoid using plain text credentials in your scripts, do one of the following:

- Require them as command-line arguments.
- Prompt the user to enter them.
- Create and use an encrypted password file.

Create encrypted password file for previously used credentials:

```
connect(username, password, myurl)  
storeUserConfig()
```

Stored in OS user's home directory by default

Connect to a server using the current password file:

```
connect(url=myurl)  
...
```

File retrieved from the home directory by default



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## WLST Best Practice: Password Management

The `storeUserConfig` command creates a user configuration file and an associated key file for the identity of the current user you are connected to WLS with. The user configuration file contains an encrypted username and password. The key file contains a secret key that is used to encrypt and decrypt the username and password. Only the key file that originally encrypted the username and password can be used to decrypt the values. If you lose the key file, you must create a new user configuration and key file pair. If you do not specify file names or locations, the command stores the files in your home directory as determined by your Java Virtual Machine (JVM). The location of the home directory depends on the SDK and type of operating system on which WLST is running. The default file names are `<username>-WebLogicConfig.properties` and `<username>-WebLogicKey.properties`.

If you do not specify credentials as part of the `connect` command, and a user configuration and a default key file exist in your home directory, then the command uses the credentials found in those files. This option is recommended if you use WLST in script mode, because it prevents you from storing unencrypted user credentials in your scripts. Alternatively, you can provide the specific locations and names of these files by using the `userConfigFile` and `userKeyFile` command arguments.

## WLST Best Practice: Error Handling

- Before reading, updating, or deleting resources, confirm that they exist.
- Before creating new resources, confirm that they do not already exist.

```
try:  
    cd('/JDBCSystemResources/' + dsName)  
    print 'The JDBC Data Source ' + dsName + ' already exists.'  
    exit()  
except WLSTException:  
    pass  
  
jdbcSystemResource = create(dsName, 'JDBCSystemResource')  
  
ds = getMBean('/JDBCSystemResources/' + dsName)  
if ds != None:  
    ...
```



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### WLST Best Practice: Error Handling

The first example shows the use of a `try/except` block as part of creating a new server resource, such as a Java Database Connectivity (JDBC) data source. The `try/except` block is used while attempting to navigate to a specific configuration MBean with the `cd` command. If this code block succeeds without an exception, the MBean already exists and the script is terminated. If an exception is thrown (the expected case), the rest of the script is executed to create the new resource. The Python `pass` statement is a required placeholder, although it performs no actual work.

The second example shows how similar functionality can be accomplished using the `getMBean` command. Unlike the `cd` command, `getMBean` does not throw an exception if the given MBean is not found.

The use of `print` statements to inform the user of script progress and successful completion is also a good practice.

# Working with Templates Using WLST

- Commands are available for editing existing templates and creating new ones.
- Edit a template's configuration by using standard offline commands such as cd, create, and set.

Command	Description
<code>readTemplate</code>	Opens an existing domain template to edit it or to create a new domain from it
<code>writeTemplate</code>	Writes the current domain configuration to a template file
<code>addTemplate</code>	Applies a given extension template to the current domain configuration
<code>loadDB</code>	Runs the SQL scripts found in the current domain for the specified database using the specified data source



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Working with Templates Using WLST

Without connecting to a running WebLogic Server instance, you can use WLST to create domain templates, create a new domain based on existing templates, or extend an existing, inactive domain. You cannot use WLST offline to view performance data about resources in a domain or modify security data (such as adding or removing users). WLST offline provides read and write access to the configuration data that is persisted in the domain's config directory or in a domain template JAR created by using the Template Builder.

Unlike the Template Builder, the `writeTemplate` command does not directly support the addition of new files and/or SQL scripts to a template JAR. However, you can use standard Jython I/O operations to add these files to an existing domain directory before creating a template from it.

WLST's `createDomain` and `setOption` commands together provide similar functionality to the Configuration Wizard. But unlike the wizard, the `createDomain` command does not support the execution of custom SQL scripts or a final.py script packaged within a domain template. To run any SQL scripts using WLST, use the `loadDB` command instead.

## Template WLST Examples

Create a domain template from an existing domain:

```
readDomain('/home/oracle/domains/mydomain')
writeTemplate('/home/oracle/templates/mydomain.jar')
```

Apply extension templates to an existing domain:

```
readDomain('/home/oracle/domains/mydomain')
addTemplate('/home/oracle/templates/testJMSMod.jar')
addTemplate('/home/oracle/templates/testApp.jar')
writeDomain('/home/oracle/domains/mydomain')
```

Create a domain from a template and run its SQL scripts:

```
readTemplate('/home/oracle/templates/mydomain.jar')
writeDomain('/home/oracle/domains/mydomain')
loadDB('10g','myOracleDataSource')
```



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Template WLST Examples

When you open a template or domain, WLST is placed at the root of the configuration hierarchy for that domain. WebLogic Server configuration beans exist within a hierarchical structure. In the WLST file system, the hierarchies correspond to drives, types and instances are directories, and attributes and operations are files. WLST traverses the hierarchical structure of configuration beans by using commands such as `cd`, `ls`, and `pwd` in a similar way that you would navigate a file system in a UNIX or Windows command shell. After navigating to a configuration bean instance, you interact with the bean using WLST commands.

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

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

## Pack Tool

- The WL\_HOME/common/bin/pack command-line tool:
  - Performs equivalent functionality to the WLST readDomain and writeTemplate commands
  - Provides a simple and convenient way of creating a domain template
- Create a template from an existing domain using pack:

```
pack.sh -domain=domains/mydomain  
        -template=templates/mydomain.jar  
        -template_name="My Custom Template"
```

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Pack Tool

The pack command provides an alternative method for creating a template from the command line in one simple step. Although the pack command does not allow you to customize the contents of your template in the same way as the Domain Template Builder, it is useful for performing the following tasks quickly:

- Creating a domain template that contains a snapshot of an entire working domain. You can then use this template as the basis for a new domain that you create by using the unpack command, Configuration Wizard, or WLST.
- Creating a managed server template that contains a subset of the files in a domain that are required to create a managed server domain directory hierarchy on a remote machine. You can then create the managed server domain directory on the remote machine by using the unpack command.

# Template Types

Type	Description	Create Using
Domain template	Defines a complete domain configuration and includes all supporting files	Template Builder, WLST, Pack
Extension template	Defines a set of domain resources, files, and applications to add to an existing domain	Template Builder
Managed server template	Includes only those domain files required to run a managed server on a separate machine from the administration server	Pack



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Template Types

The process for creating an extension template is similar to the process for creating a domain template, except that you are not prompted to configure the administration server, assign an administrator username and password, or specify the Start menu entries, because these domain settings are already defined in the domain to be extended.

A managed server template defines the subset of resources within a domain that are required to create a managed server domain directory on a remote machine. You can create a managed server template by using the `pack` command. Subsequently, when you use the managed server template with the `unpack` command, the managed server domain directory that is created contains sufficient bootstrap information to start the managed server on the remote machine. You can start the managed server on the remote machine by using either the Node Manager or customized start scripts created when you unpacked the template on the remote machine.

# Creating Managed Server Templates

- Managed server templates contain the minimal resources required to start a managed server and connect to a remote Administration Server:
  - Start scripts
  - Files required for secure sockets layer (SSL)
  - Domain configuration (to support starting independently of administration server)
- Create a managed server template from an existing domain using pack:

```
pack.sh -domain=domains/mydomain  
        -template=templates/mydomain_managed.jar  
        -template_name="My Managed Server Template"  
        -managed=true
```



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Creating Managed Server Templates

The following domain files and directories are included by default in a managed server template:

- All the files in the root directory with the following extensions: .cmd, .sh, .xml, .properties, and .ini
- Any files with the .pem extension defined in the SSL configuration for your domain
- The /bin directory
- The /lib directory
- All files and subdirectories in the config directory

The following files and directories are not included in a managed server template by default:

- Applications and certain application initialization files
- Temporary files that are created when you start a server
- The /servers directory
- Files in the /security directory that are created automatically when you create the domain, such as DefaultAuthenticatorInit.ldift and XACMLRoleMapperInit.ldift

## **Creating Managed Server Templates (continued)**

The config.xml file of the domain from which you are creating your template must contain managed server definitions that specify the IP addresses and ports for the target remote machines. The managed server template can only be used on these remote machines to create managed server domain directories.

## Unpack Tool

- The `WL_HOME/common/bin/unpack` command-line tool:
  - Performs equivalent functionality to the WLST `readTemplate` and `writeDomain` commands
  - Provides a simple and convenient way of creating a new domain from a template JAR
- Create a domain from an existing template using `unpack`:

```
unpack.sh -template=templates/mydomain.jar  
          -domain=domains/mydomain  
          -server_start_mode=prod
```



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Unpack Tool

The `unpack` command provides a one-step method for creating a domain quickly from an existing template by using the default settings defined in the template. When creating a domain, the `unpack` command does not provide the same customization options as the Configuration Wizard. If, however, you use the `unpack` command with a domain template, you can do the following:

- Change the password for the default administrative user defined in the template.
- Add an administrative user when the default administrative user already has a password specified in the template.
- Specify the JDK and start mode for the domain.
- Specify an applications directory, if one is supported by the template.

When you use the `unpack` command with a managed server template, it creates a managed server domain directory. An entry for the managed server domain directory is also created in the `NM_HOME/nodemanager.domains` file, where `NM_HOME` designates the Node Manager installation directory for the product installation on the remote machine. By default, this directory is located in `WL_HOME/common/nodemanager`.

## ConfigToScript

- The configToScript WLST command:
  - Takes an existing domain as input
  - Generates a WLST script that connects to the domain and re-creates the entire configuration
  - Creates a corresponding property file for commonly changed parameters, such as the domain name, server name, port, and administrative password
  - Creates a separate encrypted file to store any encrypted password attributes
- Use WLST to generate a domain bootstrap script:

```
configToScript('/home/oracle/domains/mydomain',
               'init_mydomain.py')
```



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## ConfigToScript

This WLST command converts an existing domain configuration (/config directory) into an executable WLST script. You can then use the resulting script to help re-create the domain resources on other machines, or restore a domain to a previously known state.

The command can be run while the domain is running or offline. It generates the following files:

- A WLST script that contains the commands needed to re-create the configuration
- A properties file that contains domain-specific values, including the location of the administration server and the required credentials to access it. You can update the values in this file to update another domain with the same configuration.
- A user configuration file and an associated key file to store encrypted attributes, such as passwords. The user configuration file contains the encrypted information. The key file contains a secret key that is used to encrypt and decrypt the encrypted information.

## Generated Script Internals

The generated script performs the following steps:

1. Start the administration server if it is not already running.
2. Connect to the administration server.
3. Create all domain resources if they do not already exist.
4. Deploy all applications if they are not already deployed.
5. Shut down the server if the script started it.



ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Generated Script Internals

The script file generated by the `configToScript` command will first try to connect to a running server using the values in the properties file. If there is no server running, WLST starts a server with the values in the properties file. You can change these property values if you want to reuse this script on another domain running in a different environment. After the script performs its changes, the server will be shut down if it was initially offline.

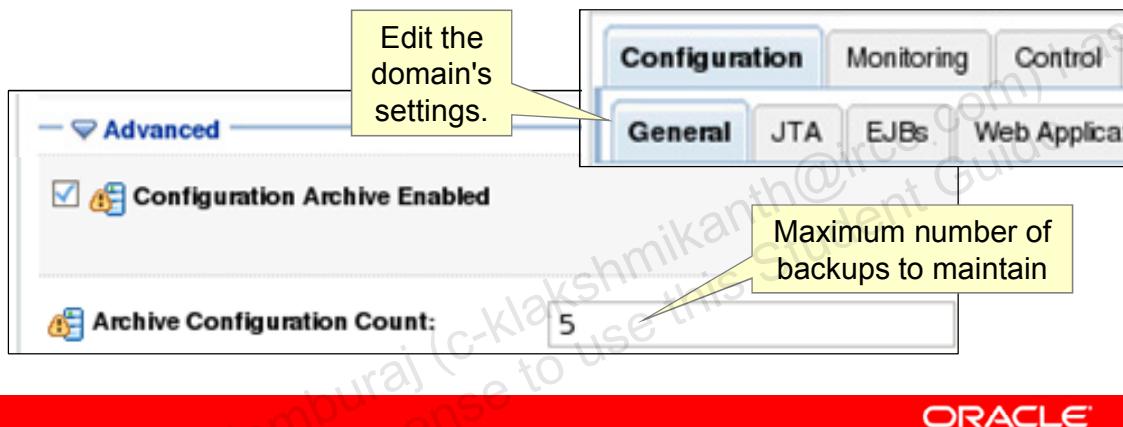
The implementation of the script includes a main function that calls several generated subroutines, which include:

- `initConfigToScriptRun`: Start the server if not already running and connect to it.
- `startTransaction`: Create an edit session on the domain.
- `create_xxx`: Create a new MBean of type XXX in the domain's configuration, if not already present.
- `setAttributesFor_xxx`: Initialize the attributes for the MBean named XXX.
- `endTransaction`: Activate the edit session.
- `deploy_xxx`: Deploy the application named XXX to the domain.

# Domain Configuration Archives

When configuration archives are enabled for a domain, the administration server:

- Creates a backup copy (JAR) of the domain's configuration each time the server is started
- Creates another backup copy each time the configuration is modified



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Domain Configuration Archives

You can configure WebLogic Server to make backup copies of the configuration files. This facilitates recovery in cases where configuration changes need to be reversed or the unlikely case that configuration files become corrupted or erroneous. When the Administration Server starts up, it saves a JAR file named `config-booted.jar` that contains the configuration files. When you make changes to the configuration files, the old files are saved in the `configArchive` directory under the domain directory, in a JAR file with a sequentially numbered name (for example, `config-1.jar`).

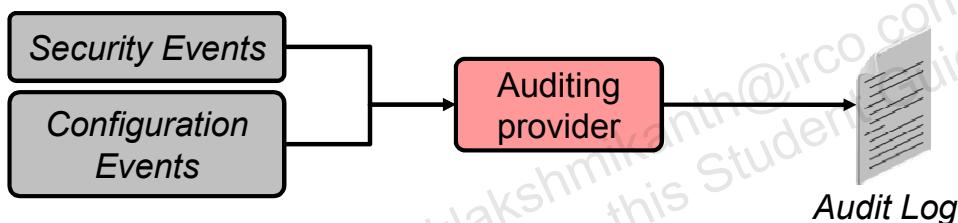
To enable and configure the configuration archive feature:

- In the left pane of the console, select the domain name. Confirm that the Configuration > General tab is selected.
- Click Advanced.
- Select the Configuration Archive Enabled check box.
- Optionally, configure the Archive Configuration Count field. By default, there is no limit on the number of configuration archive files (JARs) that an Administration Server can maintain on the file system.
- Click Save.

# WebLogic Auditing Provider

The WebLogic Auditing Provider:

- Creates a detailed record of all security changes and decisions within a domain (`DefaultAuditRecorder.log`)
- Can also create a record of all domain configuration changes
- Is not enabled by default



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## WebLogic Auditing Provider

Auditing is the process whereby information about operating requests and the outcome of those requests are collected, stored, and distributed for the purposes of nonrepudiation. In WebLogic Server, an Auditing Provider provides this electronic trail of computer activity. WebLogic Server includes a sample Auditing Provider, but, by default, it is not activated for a new security realm. The WebLogic Auditing Provider records information from a number of security requests, which are determined internally by the WebLogic Security Framework. The WebLogic Auditing Provider also records the event data associated with these security requests and the outcome of the requests.

You can also configure the Administration Server to emit audit messages that enable tracking of configuration changes in a domain. This provides a record of changes made to the configuration of any resource within a domain, as well as invocations of management operations on any resource within a domain. Configuration audit records can be saved to a log file, sent to an Auditing Provider in the security realm, or both.

All auditing information recorded by the WebLogic Auditing Provider is saved in `<domain>/servers/<server>/logs/DefaultAuditRecorder.log` by default. Although an Auditing Provider is configured per security realm, each server writes auditing data to its own log file in the server directory.

## Security Audit Events

- Typical security events include:
  - An authentication or identity assertion attempt
  - A new role or policy
  - A locked/unlocked user account
- Security events have the following characteristics:
  - Name
  - Severity (Warning, Error, Success, and so on)
  - Zero or more context attributes:
    - Protocol, port, address
    - HTTP headers
    - EJB method parameters
    - SAML tokens



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Security Audit Events

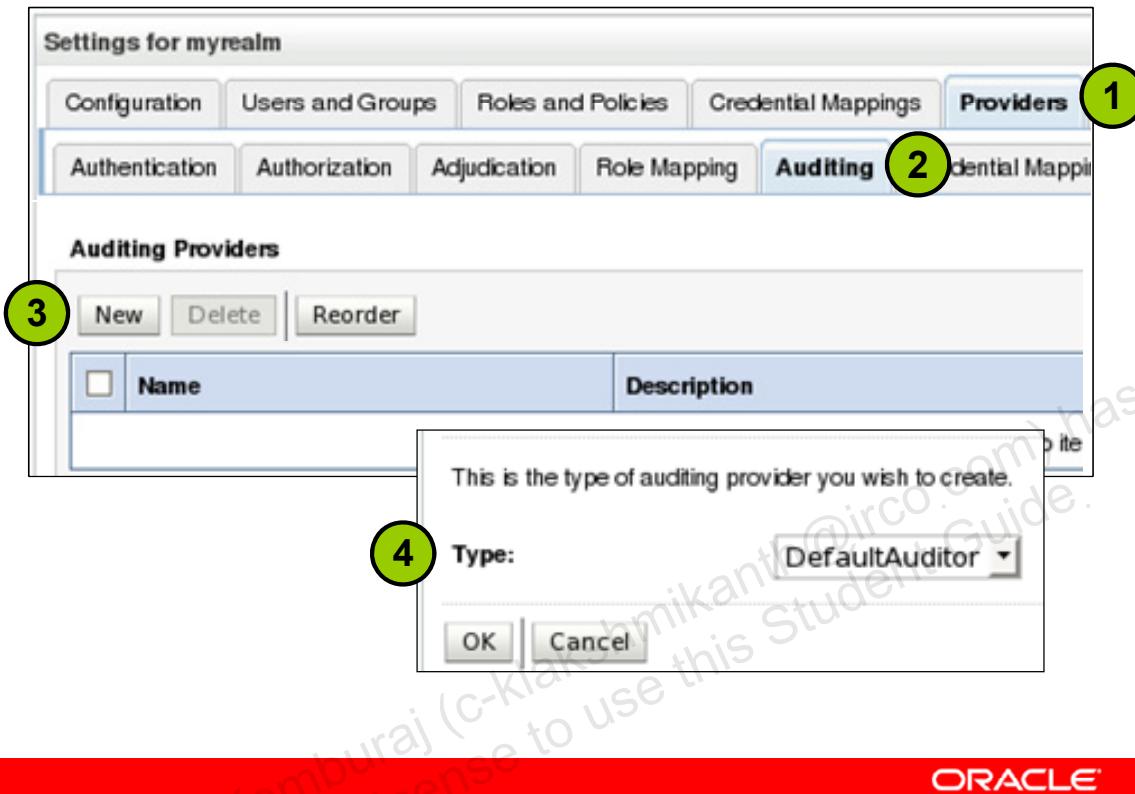
In addition to those mentioned in the slide, the default WebLogic Auditing Provider records these types of security events:

- The lock on a user account expires.
- A security policy is used and an authorization decision is made.
- A role definition is used.
- A role or policy is removed or “undeployed.”

The WebLogic Auditing Provider audits security events of the specified severity, as well as all events with a higher numeric severity rank. For example, if you set the severity level to ERROR, the WebLogic Auditing Provider audits security events of severity level ERROR, SUCCESS, and FAILURE. You can also set the severity level to CUSTOM, and then enable the specific severity levels you want to audit, such as ERROR and FAILURE events only.

An audit event includes a context object that can hold a variety of attributes, depending on the type of event. When you configure an Auditing Provider, you specify which context attributes are recorded for each event. By default, no context attributes are audited.

## Adding an Auditing Provider



ORACLE

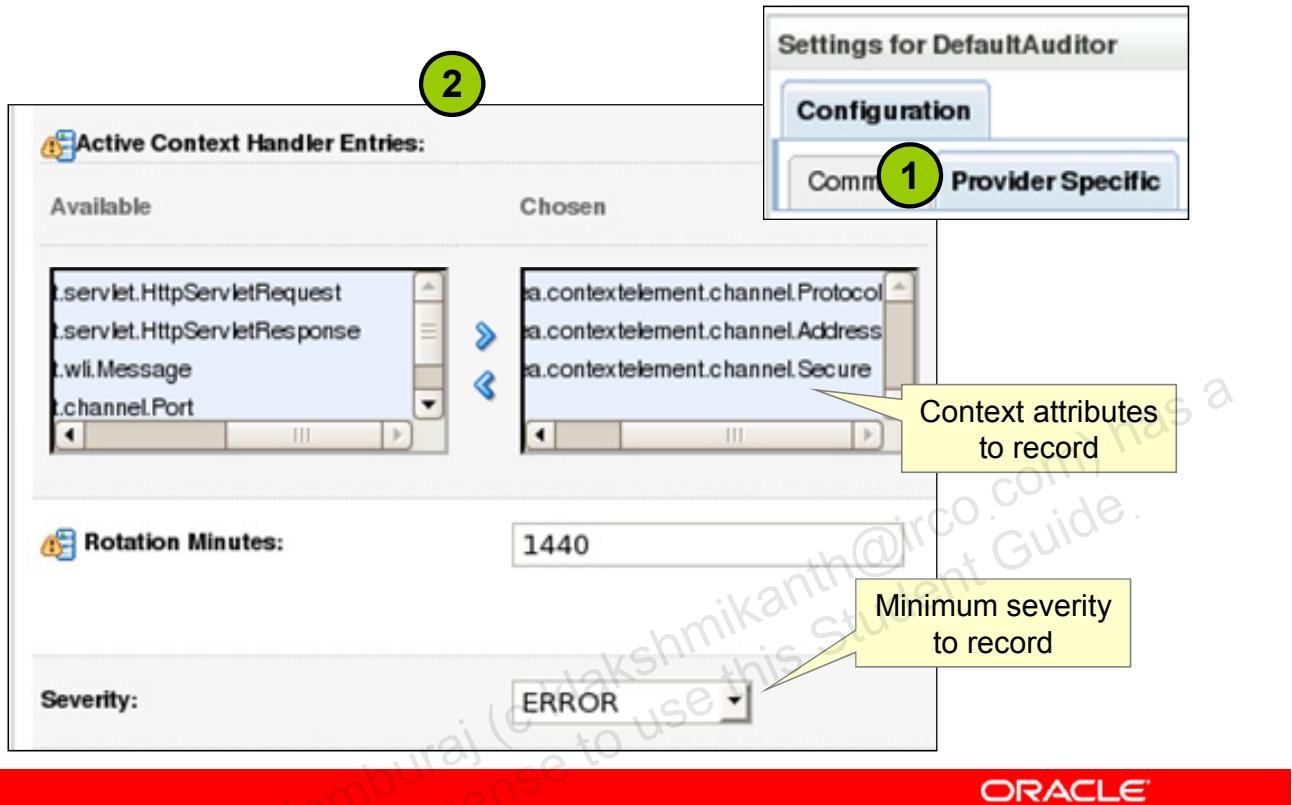
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Adding an Auditing Provider

To enable the default WebLogic Auditing Provider, perform the following steps:

1. In the left pane, select Security Realms and click the name of the realm you are configuring (for example, `myrealm`). Then click the Providers tab in the right pane.
2. Click the tab for Auditing providers.
3. Click New.
4. In the Name field, enter a name for the Auditing Provider. From the Type drop-down list, select DefaultAuditor, and click OK.

# Configuring the Auditing Provider



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

ORACLE

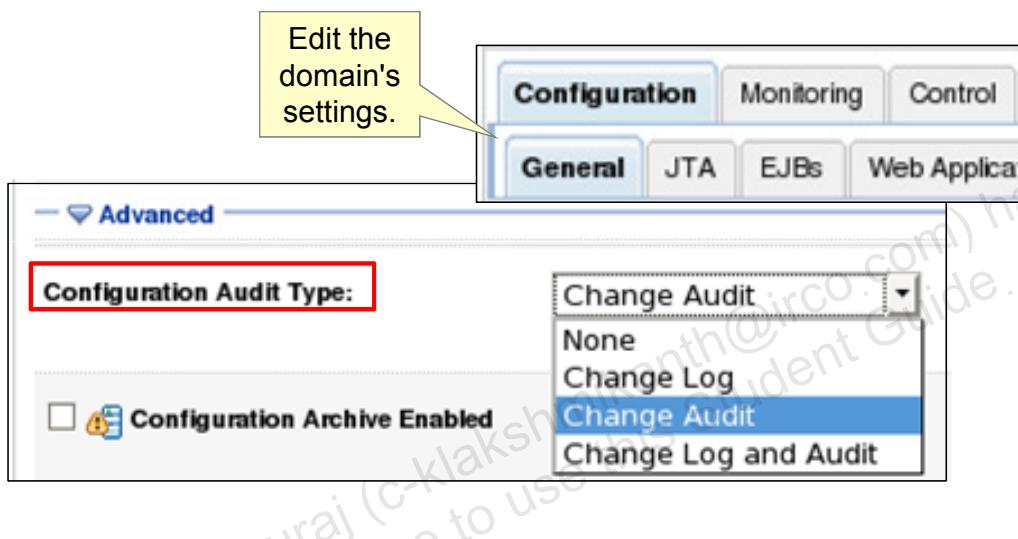
## Configuring the Auditing Provider

Select the new provider and click the Configuration > Provider-Specific tab. Optionally, update the following fields:

- **Active Context Handler Entries:** Specify which context attributes are recorded by the Auditing Provider, if present within an event. Use the arrow buttons to move the Available entries to the Chosen list.
- **Rotation Minutes:** Specifies how many minutes to wait before creating a new audit log file. After this time has elapsed, the audit file is closed and a new one is created.
- **Severity:** The minimum severity level of an event to record

## Configuration Audit Events

- Configuration auditing is enabled separately.
- Events can be sent to the Auditing Provider and/or the server log file.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Configuration Audit Events

You can configure the Administration Server to emit log messages and generate audit events when a user changes the configuration of any resource within a domain or invokes management operations on any resource within a domain. For example, if a user disables SSL on a Managed Server in a domain, the Administration Server emits log messages. If you have enabled the WebLogic Auditing Provider, it writes the audit events to an additional auditing log. These messages and audit events provide an audit trail of changes within a domain's configuration.

These audit events use MBean object names to identify resources that have been created, deleted, updated, or invoked. Object names for WebLogic Server MBeans reflect the location of the MBean within the hierarchical data model. To reflect the location, object names contain name/value pairs from the parent MBean. For example, the object name for a server's LogMBean might be `mydomain:Name=myserverlog,Type=Log,Server=myserver`.

Configuration audit events are also assigned one of three severity levels: Success, Failure, and Error.

# Quiz

What tool is used to create managed server templates?

- a. Template Builder
- b. Configuration Wizard
- c. ConfigToScript
- d. Pack



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

**Answer: d**

## Quiz

What WLST command is used to apply an extension template to an existing domain?

- a. extendDomain
- b. addTemplate
- c. importToDomain
- d. applyAll

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

**Answer: b**

## Summary

In this lesson, you should have learned how to:

- Use exception handling in WLST scripts
- Use WLST to create and update domain templates
- Use the pack/unpack tools to quickly create new templates and domains
- Rapidly create a WLST script that initializes an entire domain's configuration
- Enable automatic domain configuration backups
- Audit domain security decisions
- Track domain configuration changes



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Practice 4-1: Work with Templates from the Command Line

This practice covers the following topics:

- Opening a domain template by using WLST
- Updating an offline domain's configuration by using WLST
- Generating a template file by using WLST

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

ORACLE®

Unauthorized reproduction or distribution prohibited. Copyright© 2011, Oracle and/or its affiliates.

LakshmiKanth Kemburaj (c-klakshmikanth@irc0.com) has a  
non-transferable license to use this Student Guide.

## Advanced Network Configuration

5

ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

# Objectives

After completing this lesson, you will be able to:

- Discuss the advantages of network channels
- Configure and monitor network channels
- Create an administration channel
- Start a server in standby mode

## Default WebLogic Networking

By default, an instance of WebLogic Server binds:

- To all available network interfaces on the host machine
- To a single port
- Optionally to a separate port for secure sockets layer (SSL) communication



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Default WebLogic Networking

For many development environments, configuring WebLogic Server network resources is simply a matter of identifying a server's listen address, listen port, and optionally an SSL port. You can configure this address, port, and SSL port by using the server's Configuration > General page in the Administration Console. The values that you assign are stored in the attributes of ServerMBean and SSLMBean.

If your server instance runs on a multihomed machine and you do not configure a listen address, the server binds the listen port and/or SSL listen ports to all available IP addresses on the multihomed machine.

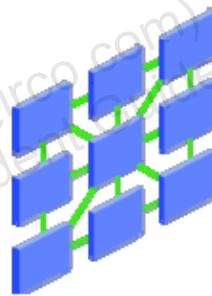
This default configuration may meet your needs if:

- Your application infrastructure has simple network requirements, such as in a development or test environment
- Your server uses a single network interface, and the default port numbers provide enough flexibility for segmenting network traffic in your domain

# Default WebLogic Networking

The default server port(s):

- Accepts all protocols (HTTP, T3, IIOP, SNMP)
- Supports various security and tuning parameters
- Is used for client-server communication
- Is used for remote server management (console, WLST)
- Is used for internal server-server communication:
  - Server startup and management messages
  - Cluster messages



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Default WebLogic Networking (continued)

You can configure each WebLogic Server instance to communicate over a variety of protocols, such as HTTP, Hypertext Transmission Protocol, Secure (HTTPS), and Internet Inter-ORB Protocol (IIOP). In addition, you can configure general server network settings that apply to all protocols. The default listen address and port accept all types of incoming server communications, including:

- Web application HTTP requests
- Remote access to the server Java Naming and Directory Interface (JNDI) tree
- Remote Enterprise JavaBeans (EJB) application Remote Method Invocations (RMI)
- Simple Network Management Protocol (SNMP) polling requests
- Configuration and monitoring requests from remote management tools, such as the console or WLST
- Configuration and monitoring requests sent from the Administration Server to the managed server
- Initial startup messages sent from a managed server to the Administration Server
- Messages sent between cluster members, such as for object replication

## Additional Networking Scenarios

Customize the default WebLogic Server network configuration to handle requirements such as:

- Dedicate network interfaces to specific servers
- Use multiple ports on a single server
- Isolate administrative communication
- Isolate internal cluster communication



ORACLE®

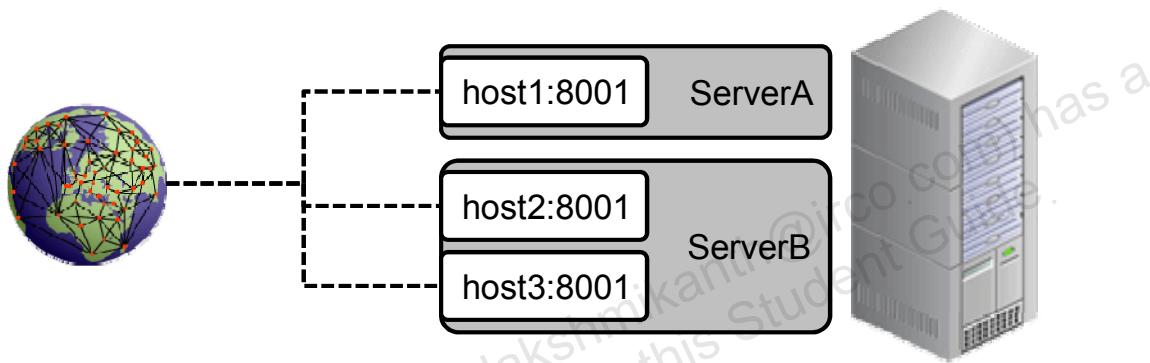
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Additional Networking Scenarios

In most production environments, administrators must balance finite network resources against the demands placed on the network. The task of keeping applications available and responsive can be complicated by specific application requirements, security considerations, and maintenance tasks, both planned and unplanned. WebLogic Server allows you to control the network traffic associated with your applications in a variety of ways and configure your environment to meet the varied requirements of your applications and end users.

## Dedicate Network Interfaces to Specific Servers

- Dedicate specific network interfaces to multiple WebLogic instances running on the same machine.
- Use the service characteristics set for each interface at the operating system level.



ORACLE®

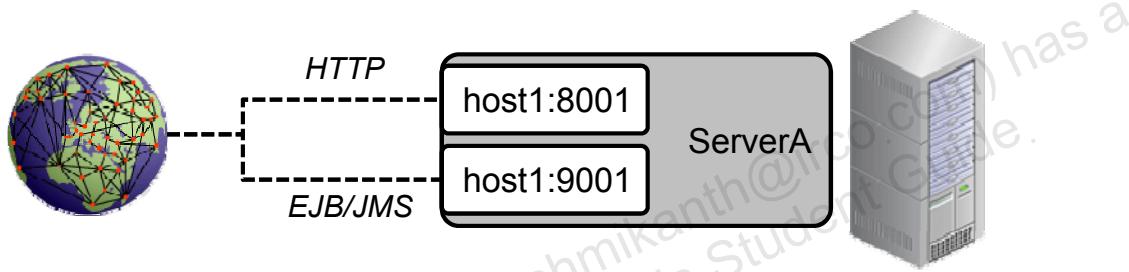
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Dedicate Network Interfaces to Specific Servers

A typical production machine hosts multiple WebLogic Server instances and also is installed with multiple network interface cards (NICs). In this scenario, it may be desirable to explicitly associate each server with its own dedicated interface or interfaces. For example, each NIC could then be tuned to support different levels of performance to match the expected load on the assigned server instance. This approach also gives administrators the flexibility to bind multiple servers on the same machine to the same port number.

## Use Multiple Ports on a Single Server

- Dedicate specific ports for certain client protocols.
- Configure different security, performance, or QoS characteristics for each port.
- Enable and disable ports independently of one another.



ORACLE®

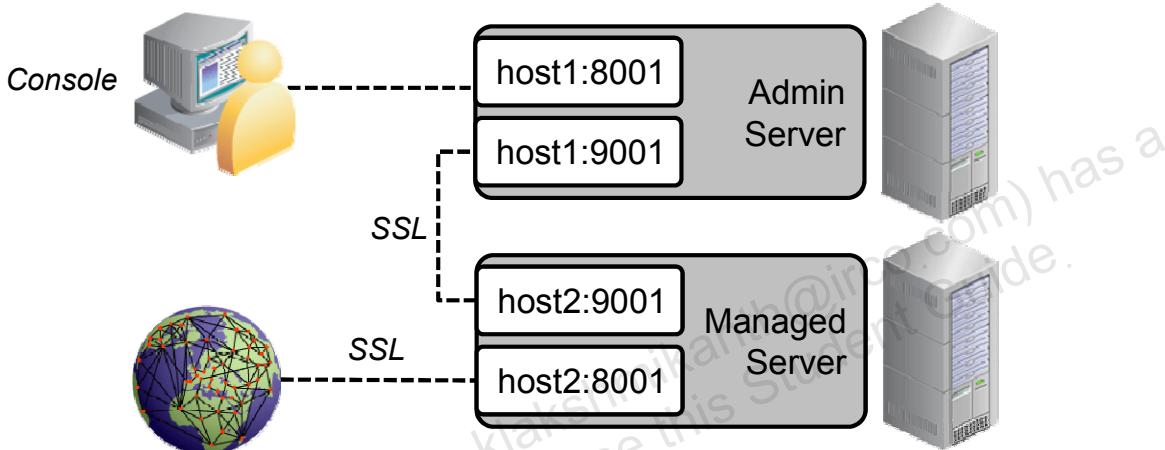
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Use Multiple Ports on a Single Server

WebLogic Server allows a single server instance to bind to multiple ports and allows administrators to define different network settings for each port. This approach lets you dedicate each port to a different protocol, such as HTTP(S) or T3(S). Also, because each port can be brought up and down independently, administrators gain additional flexibility in how they can perform server maintenance.

# Isolate Administrative Communication

- Use a dedicated address and/or port for administrative traffic.
- Disable client access points during server maintenance or troubleshooting.



**ORACLE®**

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Isolate Administrative Communication

While maintaining or troubleshooting a production server, it is often desirable to disable all incoming application requests. However, a server's default network configuration implies that all traffic runs on the same port. Therefore, if the port were closed, all remote administration tools such as the console or WLST would also not be able to connect to the server.

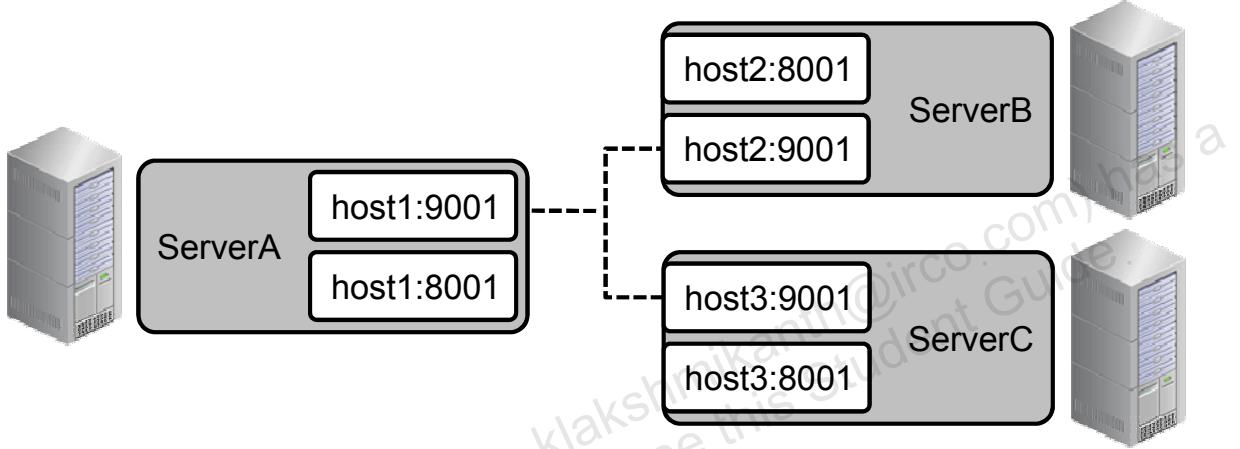
WebLogic Server supports a domain in which all servers use a separate SSL port that accepts only administration requests. The use of dedicated administration ports enables you to:

- **Start a server in standby state:** This allows you to administer a server, whereas its other network connections are unavailable to accept client connections.
- **Separate administration traffic from application traffic in your domain:** In production environments, separating the two forms of traffic ensures that critical administration operations (starting and stopping servers, changing a server's configuration, and deploying applications) do not compete with high-volume application traffic on the same network connection.
- **Administer a deadlocked server instance:** If you do not configure an administration port, administrative commands such as THREAD\_DUMP and SHUTDOWN will not work on deadlocked server instances.

## Isolate Cluster Communication

Use a dedicated address and/or port for peer-to-peer and broadcast cluster messaging:

- Server heartbeats
- Object replication



ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Isolate Cluster Communication

Similar to administration ports, the servers within a cluster can also use separate ports dedicated to internal cluster communication. Administrators have the option to configure these clusters or “replication” ports to use either a standard or a secure (SSL) protocol.

In general, ports on a server that can be used to send internal messages to other servers in the same domain are called “outgoing” ports. Ports that are not enabled for “outgoing” are used solely to process incoming client requests.

# Network Channels

- A network channel consists of a:
  - Listen address and port
  - Single supported protocol along with its service characteristics
- Each server:
  - Has an implicit default channel, which can be disabled
  - Has a default SSL channel if configured
  - Supports all protocols by default
  - Can be assigned additional channels
- Channels can be created, enabled, or disabled dynamically without restarting the server.



ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Network Channels

A network channel is a configurable resource that defines the attributes of a network connection for a specific WebLogic Server instance. A network channel definition includes a listen address, port number, supported protocol, and whether or not it can be used for internal server-to-server communication. You can use network channels to manage quality of service, meet varying connection requirements, and improve the utilization of your systems and network resources.

Administrators create a channel for a specific server instance. Channels are not defined globally and applied to one or more servers. You can assign multiple channels to a server instance, but each channel must have a unique combination of listen address, listen port, and protocol. Similarly, if you assign non-SSL and SSL channels to the same server instance, make sure that they do not use the same port number.

If you want to disable the non-SSL listen port so that the server listens only on the SSL listen port, deselect Listen Port Enabled in the Configuration > General settings for the server. Similarly, if you want to disable the SSL listen port so that the server listens only on the non-SSL listen port, deselect SSL Listen Port Enabled. Note that unless you define custom network channels, you cannot disable both the default non-SSL listen port and the SSL listen port. At least one port must be active on a server.

## Channel Selection

- For internal communication, WebLogic tries to select an available channel that best fits the requirements (supports the protocol).
- Examples include:
  - Monitoring a managed server from the Administration Server
  - Sending a cluster replication message
  - Accessing the embedded Lightweight Directory Access Protocol (LDAP)
- If multiple channels meet the criteria, messages are distributed across them:
  - Evenly (default)
  - Using channel weights



ORACLE®

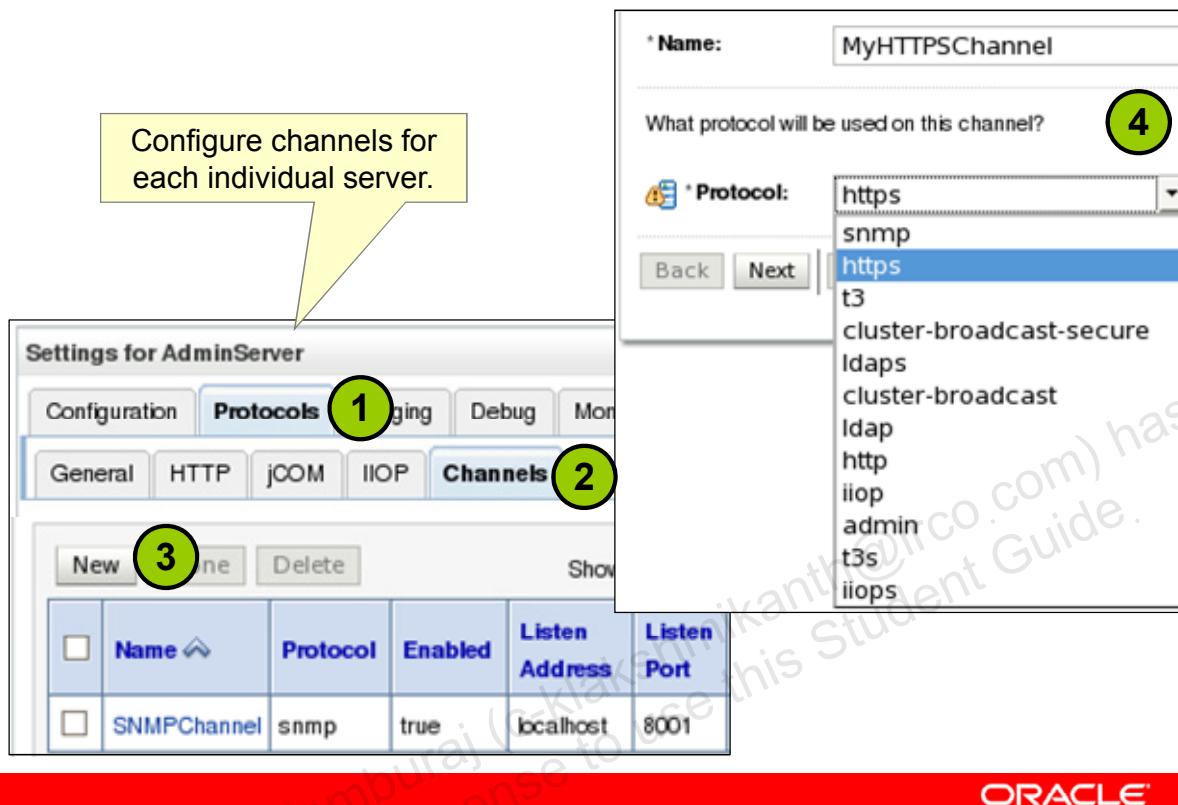
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Channel Selection

If you have not created a custom network channel to handle outgoing administrative or clustered traffic, WebLogic Server uses the default channel. If instead, multiple channels exist for these internal protocols, the server will evenly distribute the outgoing messages across the available channels. Administrators also have the option of assigning numeric weights (1–100) to internal channels for situations in which the load should not be evenly distributed.

When initiating a connection to a remote server, and multiple channels with the same required destination, protocol, and quality of service exist, WebLogic Server will try each in turn until it successfully establishes a connection or runs out of channels to try.

# Creating a Channel



ORACLE

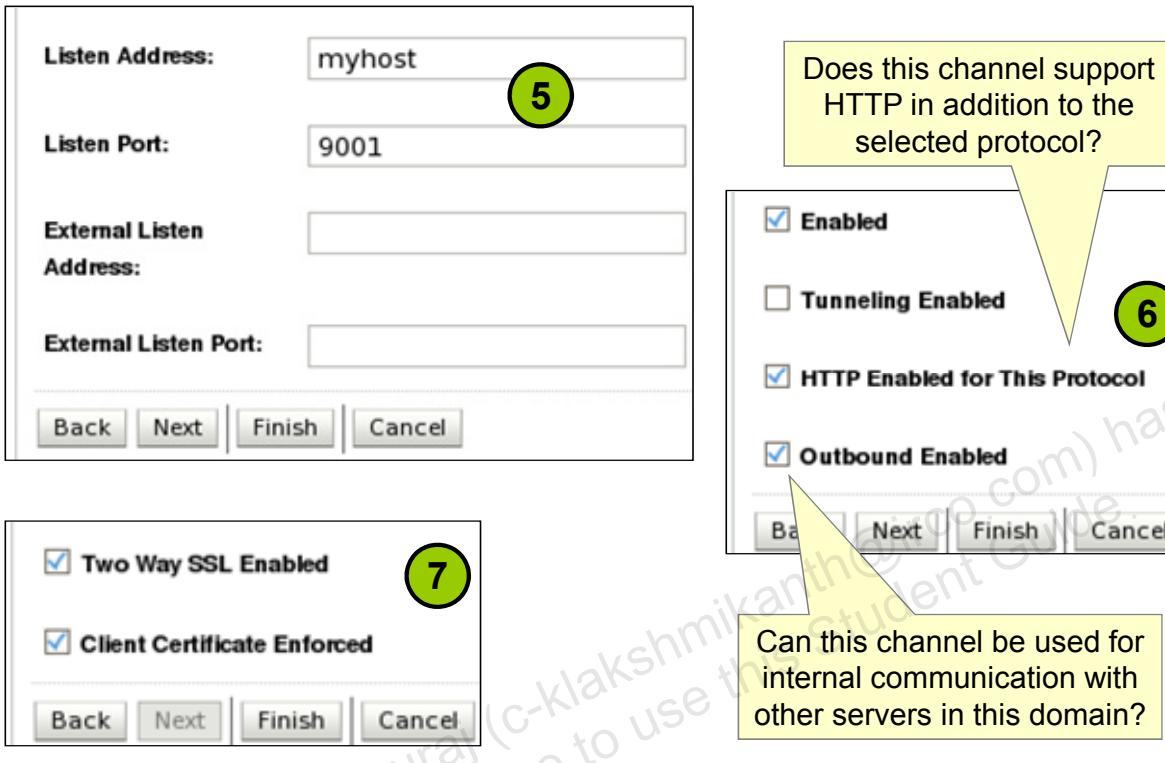
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Creating a Channel

To configure a network channel for an existing server, perform the following steps:

1. Select the server, and then click its Protocols tab.
2. Click the Channels subtab.
3. Click New.
4. Enter a name for the channel, select the protocol it will accept or use, and click Next. For administrative channels, select the admin protocol. For cluster channels, select the "cluster-broadcast" or "cluster-broadcast-secure" protocols.

# Creating a Channel



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

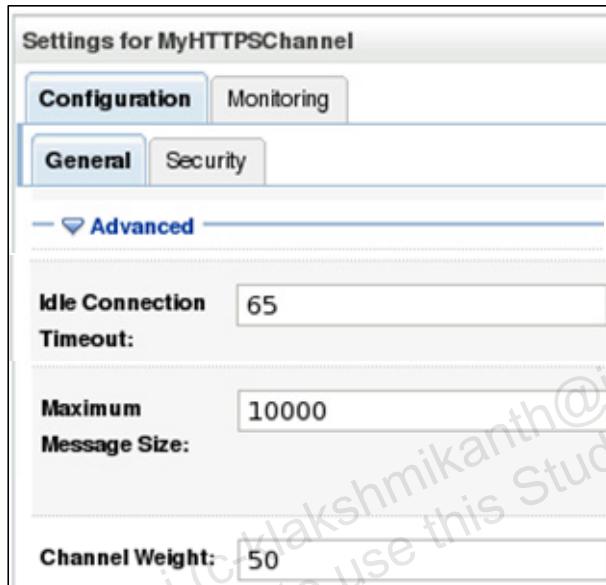
## Creating a Channel (continued)

5. Enter a listen address and listen port that this channel will bind to, and click Next. If an address is not supplied, the address of the default channel will be used.
6. Click Next. By default, the new channel will be enabled and automatically bind to its address and port. If instead you want to enable it manually at a later time, deselect the Enabled check box. Other options include:
  - HTTP Enabled for This Protocol: Specifies whether HTTP traffic should be allowed over this network channel. HTTP is generally required for other binary protocols for downloading stubs and other resources (only applicable if selected protocol is not HTTP or HTTPS).
  - Outbound Enabled: Specifies whether new server-to-server connections may consider this network channel when initiating a connection. Leave this field deselected for client channels.
7. For secure protocols, optionally enable two-way SSL. Click Finish.

If your server is being accessed through a proxy server on a separate listen address and/or port, you may be required to supply an external listen address and/or external listen port for each channel. These values will be used in cases where the server must publish its location to external clients, such as a Web server plug-in or a hyperlink in a Web browser.

# Channel Network Settings

Custom channels inherit their network settings from the default channel, if not overridden.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Channel Network Settings

Both the default and custom network channels support various general and protocol-specific network settings. If not configured, custom channels inherit their network settings from the default channel. These settings include:

- **Cluster Address:** The address this network channel uses to generate cluster-aware EJB stubs for load balancing and failover in a cluster
- **Accept Backlog:** The number of backlogged, new TCP connection requests that this network channel allows
- **Maximum Backoff Between Failures:** The maximum wait time in seconds between failures while accepting client connections
- **Idle Connection Timeout:** The maximum amount of time (in seconds) that a connection is allowed to be idle before it is closed by this network channel. This timeout helps guard against server deadlock through too many open connections.
- **Maximum Message Size:** The maximum attempts to prevent a denial of service attack whereby a caller attempts to force the server to allocate more memory than is available thereby keeping the server from responding quickly to other requests
- **Channel Weight:** A weight to give this channel when multiple channels are available for internal server-to-server connections

## Channel WLST Example

Create a custom channel by using WLST:

```
edit()
startEdit()

server = getMBean('/Servers/serverA')
channel = server.createNetworkAccessPoint('MyHTTPSChannel')
channel.setProtocol('https')
channel.setListenAddress('myhost')
channel.setListenPort(9001)
channel.setMaxMessageSize(10240)
channel.setEnabled(true)

save()
activate(block='true')
```



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

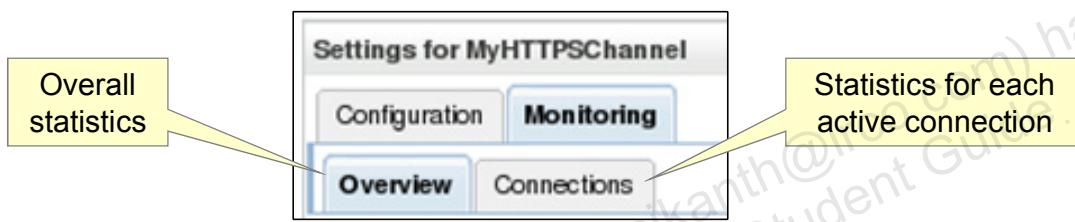
## Channel WLST Example

Refer to the NetworkAccessPointMBean for a complete list of attributes.

# Monitoring Channels

Runtime statistics are available for each channel:

- Number of Active Connections
- Messages Sent/Received
- Bytes Sent/Received



ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Monitoring Channels

1. Select a server.
2. Select Protocols > Channels.
3. Select a channel.
4. Click the Monitoring tab.

## Creating Administration Channels

- An alternative to creating an explicit “admin” protocol channel on every server is the domain’s *Administration Port setting*.
- This feature creates an implicit administration channel on every server using:
  - The listen address of the server’s default channel
  - The same port number
- Administration channels require SSL to be configured on each server.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

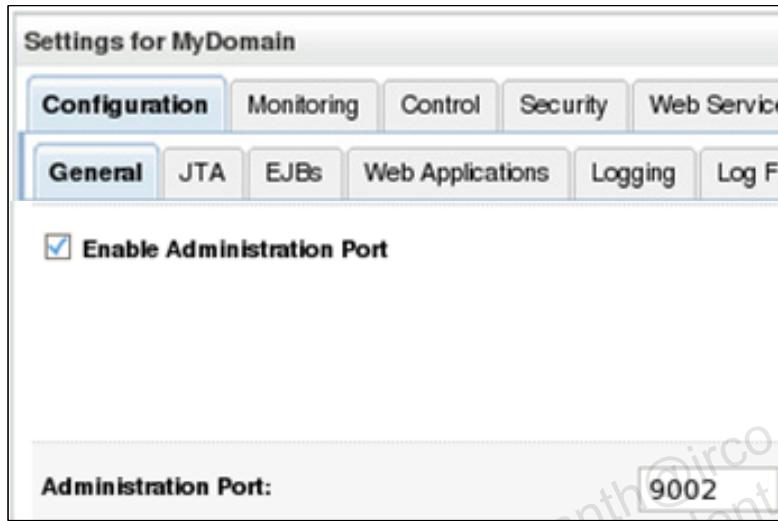
### Creating Administration Channels

You can define an optional administration port for your domain. When configured, the administration port is used by each managed server in the domain exclusively for communication with the domain’s Administration Server. If an administration port is enabled, WebLogic Server automatically generates an administration channel based on the port settings upon server instance startup.

The administration port accepts only secure, SSL traffic, and all connections via the port require authentication. The Administration Server and all managed servers in your domain must be configured with support for the SSL protocol. Managed servers that do not support SSL cannot connect with the Administration Server during startup—you will have to disable the administration port in order to configure them.

Ensure that each server instance in the domain has a configured default listen port or default SSL listen port. The default ports are those you assign on the Server > Configuration > General page in the administration console. A default port is required in the event that the server cannot bind to its configured administration port. If an additional default port is available, the server will continue to boot and you can change the administration port to an acceptable value.

# Administration Port



ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Administration Port

To use a domain-wide administration port, perform the following steps:

1. Select the domain name in the left panel.
2. On the default Configuration > General tab, select the Enable Administration Port check box.
3. Enter a value for the Administration Port field and click Save.

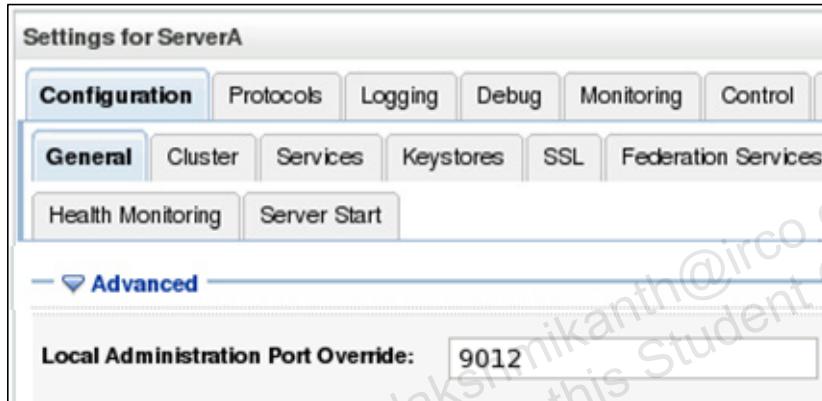
After enabling the administration port, all administration console and WLST traffic must connect via the administration port.

If you boot managed servers either at the command line or by using a start script, specify the administration port in the administration server's URL. The URL must also specify the HTTPS protocol rather than HTTP. If you use Node Manager for starting managed servers, it is not necessary to modify startup settings or arguments for the managed servers. Node Manager obtains and uses the correct URL to start a managed server.

## Administration Port Override

If multiple servers run on the same machine, you must perform either of the following:

- Bind each server to a unique network address
- Override the administration port for individual servers



ORACLE®

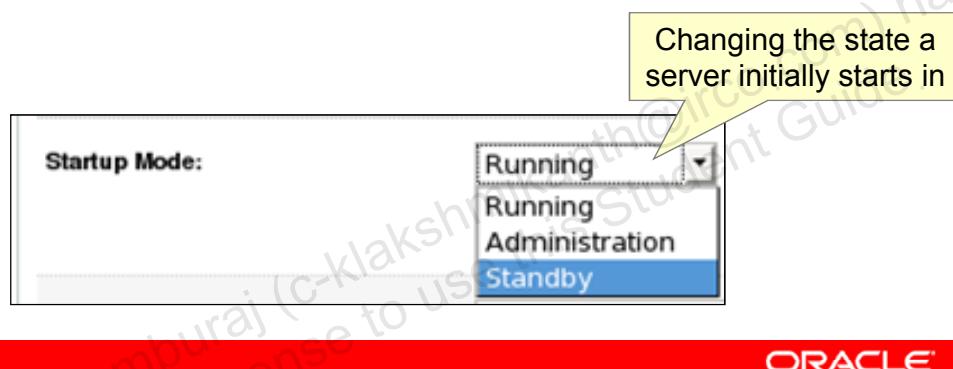
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Administration Port Override

Override the domain-wide port on all but one of the server instances running on the same machine. Override the port using the Local Administration Port Override option on the Advanced portion of the server's Configuration > General tab in the administration console.

## Server Standby Mode

- In the ADMIN state, all channels are opened but applications are only accessible to administrators.
- In the STANDBY state, only a server's administration channel is opened; all other channels remain closed.
- Administrators can later transition servers in the ADMIN or STANDBY state to the RUNNING state.



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

ORACLE

### Server Standby Mode

In the ADMIN state, WebLogic Server is up and running, but available only for administration operations, allowing you to perform server- and application-level administration tasks. The server instance accepts requests from users with the Admin role. Requests from non-admin users are refused. All applications accept requests from users with the Admin and AppTester roles. The Java Database Connectivity (JDBC), Java Message Service (JMS), and Java Transaction API (JTA) subsystems are active, and administrative operations can be performed upon them. However, you do not have to have administrator-level privileges to access these subsystems when the server is in the ADMIN state.

A server instance in STANDBY does not process any client requests; all nonadministration network channels do not open. The server's administration channel is open and accepts lifecycle commands that transition the server instance to either the RUNNING or the SHUTDOWN state. Other types of administration requests are not accepted. Starting a server instance in standby is a method of keeping it available as a "hot" backup, a useful capability in high-availability environments.

The only way to cause a server instance to enter the STANDBY state and remain in that state is through the server's Startup Mode attribute, found in the Advanced section of the Configuration > General tab.

## Quiz

Which of the following is NOT a use of network channels?

- a. Entitle application users
- b. Open multiple ports on a single server
- c. Dedicate a port to administration traffic
- d. Maintain separate settings for multiple network interfaces



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

**Answer: a**

# Quiz

Name three protocols that you can assign to a network channel.

- a. ORCL
- b. T3
- c. HTTPS
- d. VNC
- e. ADMIN

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

**Answer:** b, c, e

## Summary

In this lesson, you should have learned how to:

- Describe a server's default network characteristics
- Explain several networking scenarios that can be implemented using network channels
- Create an administration channel
- Start a server in standby mode

## Practice 5-1: Use Network Channels

This practice covers the following topics:

- Configuring dedicated administration channels for servers
- Accessing an administration server by using its administration channel
- Using an administration channel to boot a managed server
- Starting a managed server in standby mode

# Multi Data Sources

ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

# Objectives

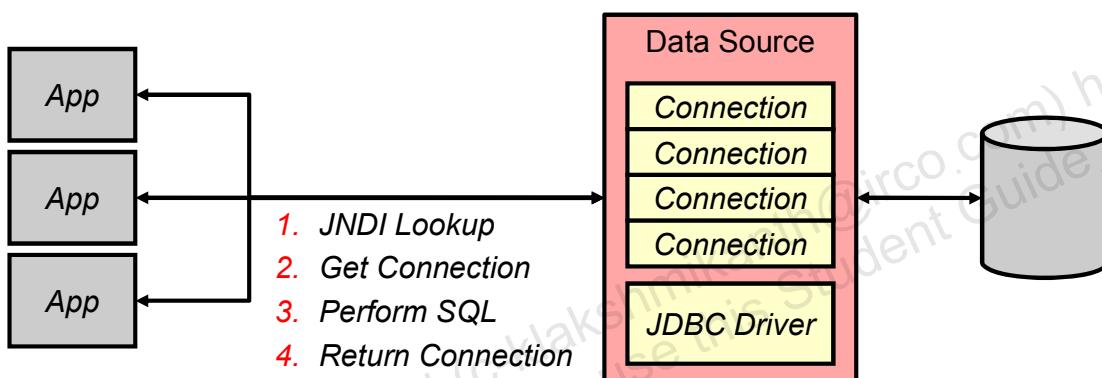
After completing this lesson, you will be able to:

- Create multi data sources to support highly available database systems
- Select a multi data source algorithm
- Describe approaches for using WLS with Oracle Real Application Clusters (RAC)

# Data Source Review

## Data sources:

- Allow database connectivity to be managed by the application server
- Are obtained by applications from the server's JNDI tree
- Use a dynamic pool of reusable database connections



**ORACLE®**

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Data Source Review

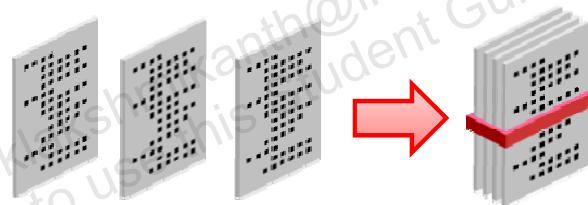
Oracle WebLogic Server can manage your database connectivity through JDBC Data Sources and multi data sources. Each data source that you configure contains a pool of database connections that are created when the data source instance is created—when it is deployed or targeted, or at server startup. The connection pool can grow or shrink dynamically to accommodate demand.

Applications look up a data source on the Java Naming and Directory Interface (JNDI) tree or in the local application context (`java:comp/env`), depending on how you configure and deploy the object, and then request a database connection. When finished with the connection, the application uses the close operation on the connection, which simply returns the connection to the connection pool in the data source.

Oracle WebLogic Server data sources allow connection information such as the JDBC driver, the database location (URL), and the username and password to be managed and maintained in a single location, without requiring the application to worry about these details. In addition, limiting the number of connections is important if you have a licensing limitation on your database or it can support only a specific capacity.

# XA Data Source Review

- Data source objects retrieved by applications via JNDI can be of one of two types:
  - Non-XA (default)
  - XA
- XA data sources:
  - Will automatically participate in distributed or “global” transactions initiated by the application
  - Typically require the underlying JDBC driver to support XA
  - Can use a non-XA driver in certain scenarios



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## XA Data Source Review

One of the most fundamental features of WebLogic Server is transaction management. Transactions are a means to guarantee that database changes are completed accurately. WebLogic Server protects the integrity of your transactions by providing a complete infrastructure for ensuring that database updates are done accurately, even across a variety of resource managers. If any one of the operations fails, the entire set of operations is rolled back.

If you use global or XA transactions in your applications, you should use an XA JDBC driver to create database connections in the JDBC data source. If an XA driver is unavailable for your database, or you prefer not to use an XA driver, you should enable support for global transactions in the data source. You should also enable support for global transactions if your applications meet any of the following criteria:

- They use the EJB container in WebLogic Server to manage transactions.
- They include multiple database updates within a single transaction.
- They access multiple resources, such as a database and the Java Messaging Service (JMS), during a transaction.
- They use the same data source on multiple servers (clustered or nonclustered).

## Multi Data Sources

- To avoid a single point of failure and to achieve greater scalability, many enterprises employ multiple database servers.
- A multi data source:
  - Is a pool of data sources
  - Is used by applications exactly like a standard data source
  - Transparently provides load balancing or failover across the member data sources
  - Can be XA or non-XA



ORACLE®

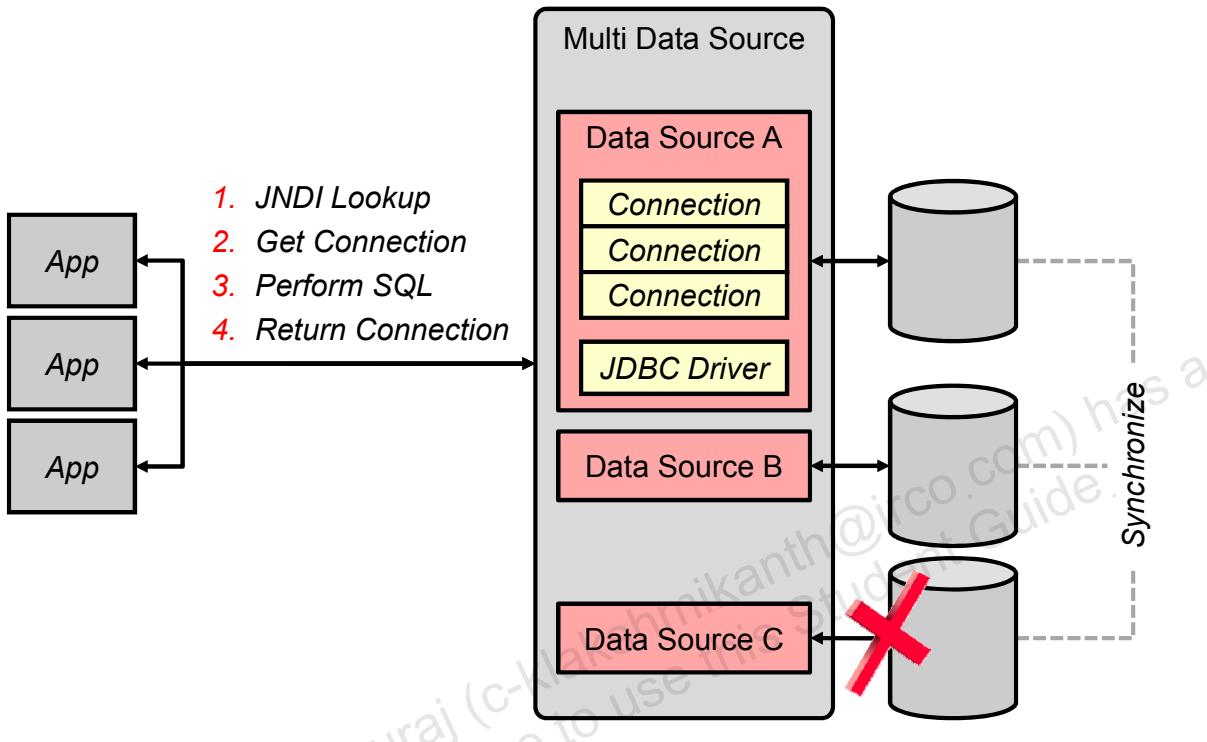
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Multi Data Sources

A multi data source is an abstraction around a group of data sources that provides load balancing or failover processing between the data sources associated with the multi data source. Multi data sources are bound to the JNDI tree or local application context just like data sources are bound to the JNDI tree. Applications look up a multi data source on the JNDI tree just like they do for data sources, and then request a database connection. The multi data source determines which data source to use to satisfy the request depending on the algorithm selected in the multi data source configuration: load balancing or failover.

All data sources used by a multi data source to satisfy connection requests must be deployed on the same servers and clusters as the multi data source. A multi data source always uses a data source deployed on the same server to satisfy connection requests. Multi data sources do not route connection requests to other servers in a cluster or in a domain. To deploy a multi data source to a cluster or server, you select the server or cluster as a deployment target. When a multi data source is deployed on a server, WebLogic Server creates an instance of the multi data source on the server. When you deploy a multi data source to a cluster, WebLogic Server creates an instance of the multi data source on each server in the cluster.

# Multi Data Source Architecture



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Multi Data Source Architecture

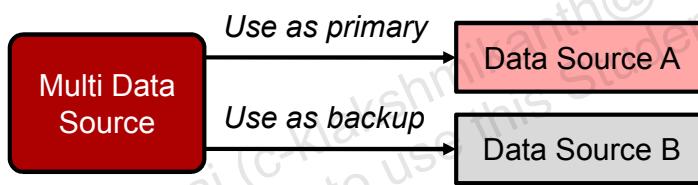
A multi data source can be thought of as a pool of data sources. Multi data sources are best used for failover or load balancing between nodes of a highly available database system, such as redundant databases or Oracle Real Application Clusters (RAC). Multi data sources do not provide any synchronization between databases. It is assumed that database synchronization is handled properly outside of WebLogic Server so that data integrity is maintained.

You create a multi data source by first creating data sources, then creating the multi data source using the administration console or the WebLogic Scripting Tool (WLST), and then assigning the data sources to the multi data source.

The data source member list for a multi data source supports dynamic updates. You can remove a database node and corresponding data sources without redeployment. This capability provides you the ability to shut down a node for maintenance or shrink a cluster.

## Failover Option

- The multi data source uses the first member data source to handle all connection requests.
- During a connection request, the other member data sources are tried in succession if the current data source:
  - Becomes unavailable
  - Has no unused connections (optional)
  - Is suspended by the administrator
- If a connection fails when in use, the application must still handle it programmatically.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

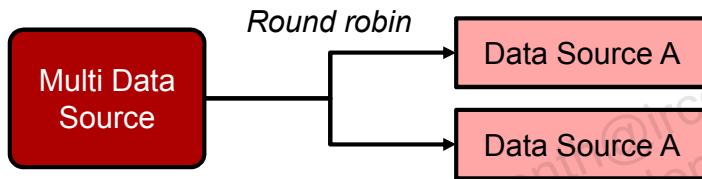
### Failover Option

The multi data source failover algorithm provides an ordered list of data sources to use to satisfy connection requests. Normally, every connection request to this kind of multi data source is served by the first data source in the list. If a database connection test fails and the connection cannot be replaced, or if the data source is suspended, a connection is sought sequentially from the next data source on the list.

JDBC is a highly stateful client-DBMS protocol, in which the DBMS connection and transactional state are tied directly to the socket between the DBMS process and the client (driver). Therefore, it is still possible for a connection to fail after being reserved, in which case your application must handle the failure. WebLogic Server cannot provide failover for connections that fail while being used by an application. Any failure while using a connection requires that the application code handle it, such as restarting the transaction.

## Load Balancing Option

- The multi data source selects member data sources to satisfy connection requests using a round-robin scheme.
- Data source failure is handled in the same fashion as described for the Failover option.



ORACLE®

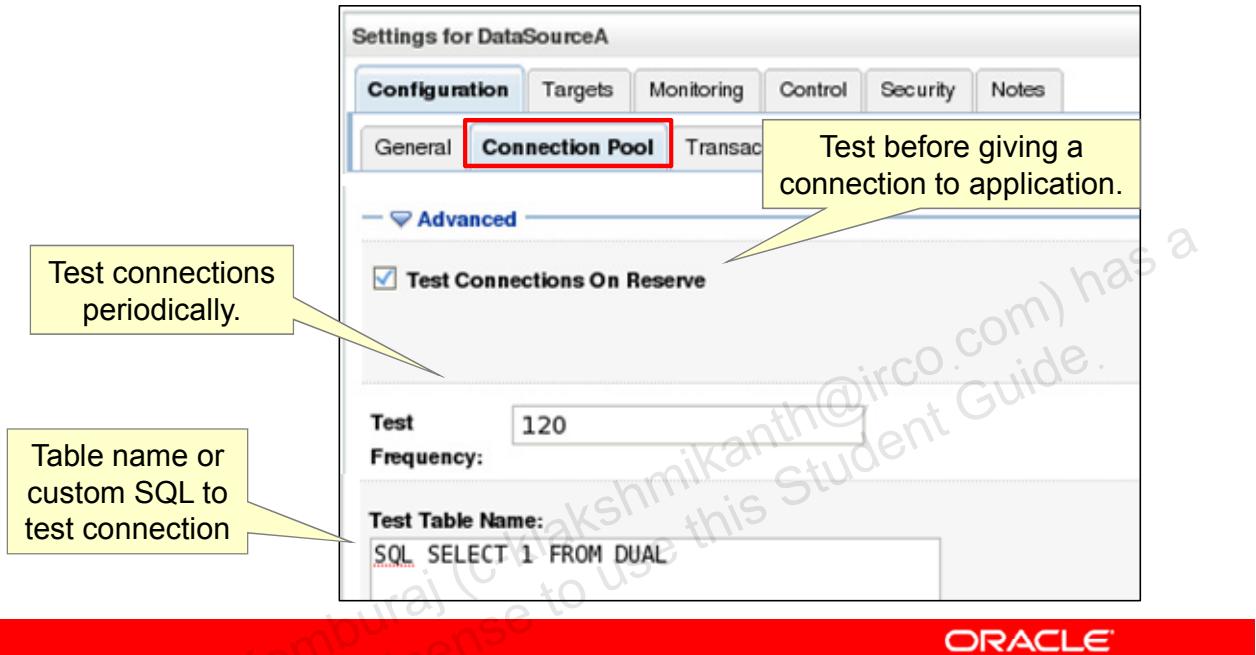
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Load Balancing Option

Connection requests to a load-balancing multi data source are served from any data source in the list. The multi data source selects member data sources to satisfy connection requests using a round-robin scheme. When the multi data source provides a connection, it selects a connection from the data source listed just after the last data source that was used to provide a connection. Multi data sources that use the load balancing algorithm also fail over to the next data source in the list if a database connection test fails and the connection cannot be replaced, or if the data source is suspended.

# Connection Testing

To provide failover, multi data sources require that all member data sources be configured to use connection testing.



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Connection Testing

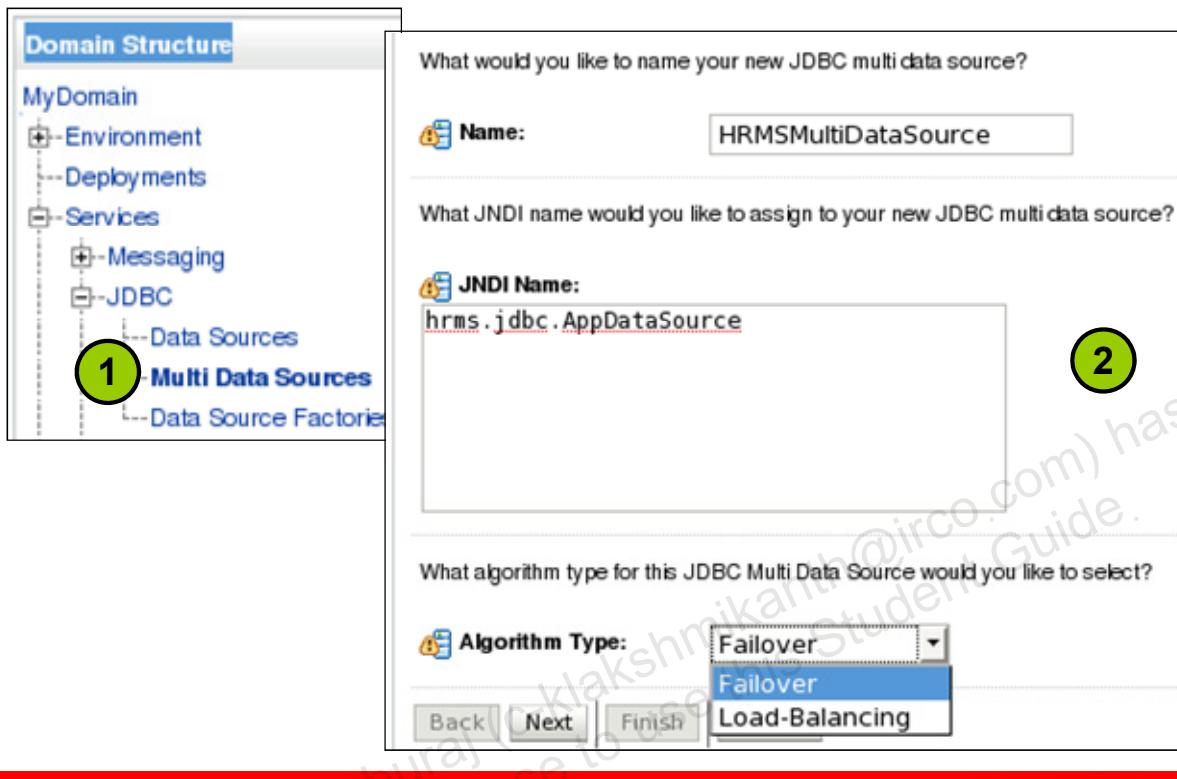
Data sources rely on the Test Reserved Connections feature to know when database connectivity is lost. Testing reserved connections must be enabled and configured for all of the data sources within the multi data source. WebLogic Server will test each connection before giving it to an application. With the failover algorithm, the multi data source uses the results from connection test to determine when to fail over to the next data source in the multi data source. After a test failure, the data source attempts to re-create the connection. If that attempt fails, the multi data source fails over to the next data source.

**Test Frequency:** Enable periodic background connection testing by entering the number of seconds between periodic tests.

**Test Reserved Connections:** Select this check box to test the database connection before giving it to your application when your application requests a connection from the data source.

**Test Table Name:** Enter the name of a small table to use in a query to test database connections. The standard query is select count(\*) from <table>. Most database servers optimize this SQL to avoid a full table scan, but it is still a good idea to use the name of a table that is known to have few rows, or even no rows. If you prefer to use a different query as a connection test, enter SQL followed by a space and the SQL code that you want to use to test database connections.

# Creating a Multi Data Source

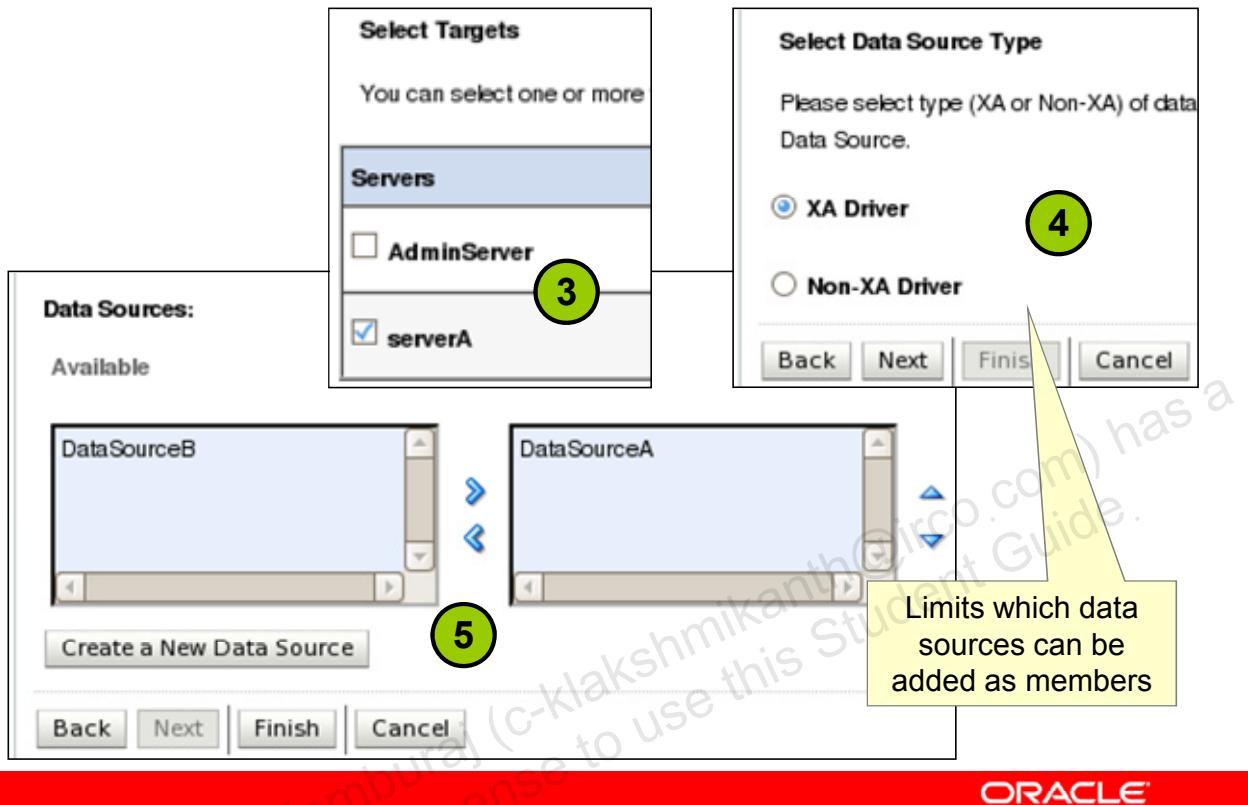


Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Creating a Multi Data Source

1. In the Domain Structure panel, expand Services > JDBC, and then select Multi Data Sources. Click New.
2. Enter or select the following information and click Next:
  - Name:** Enter a unique name for this JDBC multi data source.
  - JNDI Name:** Enter the JNDI path to where this JDBC data source will be bound. Applications look up the data source on the JNDI tree by this name when reserving a connection. To specify multiple JNDI names for the multi data source, enter each on a separate line.
  - Algorithm Type:** Select an algorithm option:
    - **Failover:** The multi data source routes connection requests to the first data source in the list; if the request fails, the request is sent to the next data source in the list, and so forth.
    - **Load-Balancing:** The multi data source distributes connection requests evenly to its member data sources.

# Creating a Multi Data Source



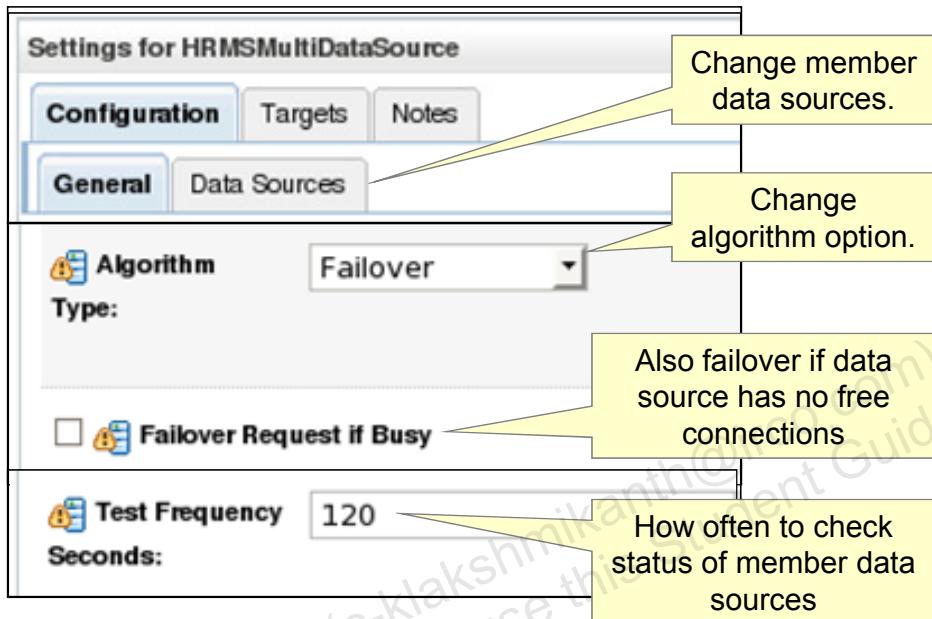
ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Creating a Multi Data Source (continued)

3. Select the servers or clusters on which you want to deploy the multi data source. The targets that you select will limit the data sources that you can select as part of the multi data source. You can only select data sources that are deployed to the same targets as the multi data source. Click Next.
4. Select one of the following options and click Next. The option that you select limits the data sources that you can select as part of the multi data source in a later step. Limiting data sources by JDBC driver type enables the WebLogic Server transaction manager to properly complete or recover global transactions that use a database connection from a multi data source:
  - **XA Driver:** The multi data source will only use data sources that use an XA JDBC driver to create database connections.
  - **Non-XA Driver:** The multi data source will only use data sources that use a non-XA JDBC driver to create database connections.
5. Select the data sources that you want the multi data source to use to satisfy connection requests. Then use the supplied arrow buttons to reorder the chosen data source list as desired and click Finish. For convenience, you can also create new data sources at this time using the "Create a New Data Source" button.

# Configuring a Multi Data Source



ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Configuring a Multi Data Source

1. In the Domain Structure panel, expand Services > JDBC, and then select Multi Data Sources.
2. Click the existing data source.
3. Edit any of the following fields and then click Save. In general, changes take effect after you redeploy the multi data source or restart the server:
  - **JNDI Name:** Same field as when creating a new multi data source
  - **Algorithm Type:** Same field as when creating a new multi data source
  - **Failover Request if Busy:** For multi data sources with the failover algorithm, enables the multi data source to fail over connection requests to the next data source if all connections in the current data source are in use
  - **Test Frequency Seconds:** The number of seconds between when WebLogic Server tests unused connections (requires that you specify a test table name). Connections that fail the test are closed and reopened to reestablish a valid physical connection. If the test fails again, the connection is closed. In the context of multi data sources, this attribute controls the frequency at which WebLogic Server checks the health of data sources it had previously marked as unhealthy. When set to 0, the feature is disabled.

## Multi Data Source WLST Example

Create a new multi data source:

```
edit()
startEdit()

jdbcSystemResource = create('HRMSMultiDataSource',
    'JDBCSYSTEMRESOURCE')
jdbcResource = jdbcSystemResource.getJDBCResource()
jdbcResource.setName('HRMSMultiDataSource')
jdbcResourceParams = jdbcResource.getJDBCDataSourceParams()
jdbcResourceParams.setJNDINames(['jdbc.hr.HRMSDS'])
jdbcResourceParams.setAlgorithmType('Failover')
jdbcResourceParams.setDataSourceList('DataSourceA',
    'DataSourceB')
jdbcSystemResource.addTarget(getMBean('/Servers/serverA'))

save()
activate(block='true')
```



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Multi Data Source WLST Example

The same JDBCSYSTEMRESOURCE MBean used to configure standard data sources is also used to configure multi data sources. This MBean has a child MBean of type JDBCDataSourceBean, which in turn has a child of type JDBCDataSourceParamsBean. Several of these parameters are only applicable to multi data sources, including:

- AlgorithmType
- DataSourceList
- FailoverRequestIfBusy

# Managing Multi Data Source Members

- To add a new database node:
  1. Create a new data source
  2. Update the multi data source and clear all target servers
  3. Update the multi data source members list
  4. Update the multi data source and retarget
- To remove a database node:
  1. Update the multi data source and clear all target servers
  2. Update the multi data source members list
  3. Update the multi data source and retarget
  4. Suspend the corresponding data source
  5. Shut down the database



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Managing Multi Data Source Members

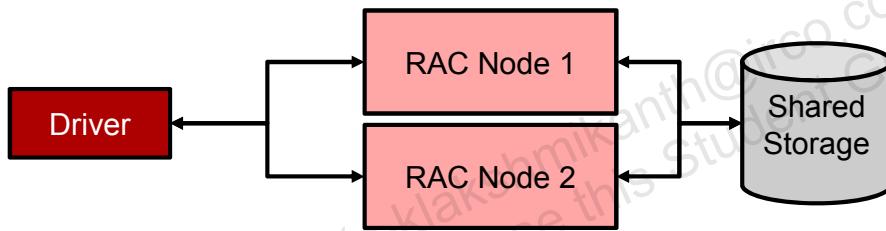
The data source member list for a multi data source supports dynamic updates. This allows environments, such as those using Oracle RAC, to add and remove database nodes and corresponding data sources without restarting the server or losing connectivity to other member data sources. For example, situations may arise in which you need to grow and shrink RAC clusters in response to throughput, or shut down RAC nodes for maintenance. You can do all of the required steps in one configuration edit session.

To improve performance when a data source within a multi data source fails, WebLogic Server automatically disables the data source when a pooled connection fails a connection test. After a data source is disabled, WebLogic Server does not route connection requests from applications to the data source. Instead, it routes connection requests to the next available data source listed in the multi data source.

After a data source is automatically disabled because a connection failed a connection test, the multi data source periodically tests a connection from the disabled data source to determine when the data source (or underlying database) is available again. When the data source becomes available, the multi data source automatically re-enables the data source and resumes routing connection requests to the data source, depending on the multi data source algorithm and the position of the data source in the list of included data sources. Frequency of these tests is controlled by the Test Frequency Seconds attribute of the multi data source.

# Oracle Real Application Clusters (RAC) Overview

- Oracle RAC:
  - Supports multiple Oracle database servers for greater scalability
  - Relies on database servers having access to a shared and highly available storage device
- The Oracle JDBC driver can provide a level of load balancing and failover across RAC instances.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

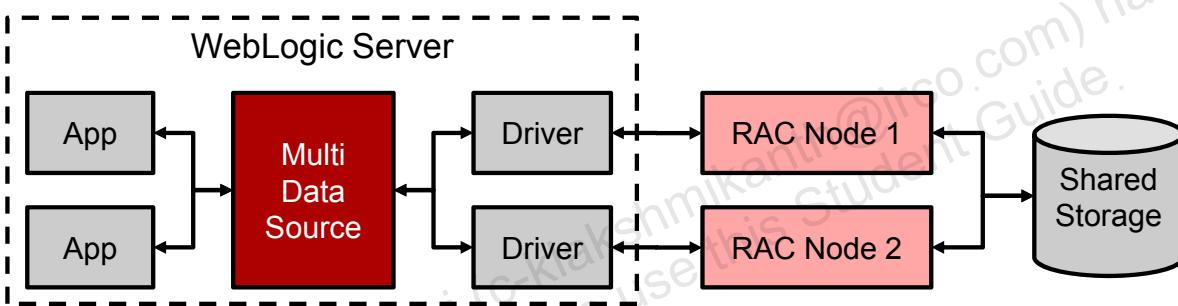
## Oracle Real Application Clusters (RAC) Overview

Oracle Real Application Clusters (RAC) is a software component you can add to a high-availability solution that enables users on multiple machines to access a single database with increased performance. RAC comprises two or more Oracle database instances running on two or more clustered machines and accessing a shared storage device via cluster technology. To support this architecture, the machines that host the database instances are linked by a high-speed interconnect to form the cluster. The interconnect is a physical network used as a means of communication between the nodes of the cluster. Cluster functionality is provided by the operating system or compatible third-party clustering software.

Depending on your configuration, when a RAC node or instance fails, session requests are redirected to another node in the cluster either by WebLogic Server or by the Oracle Thin driver. Note that with WebLogic Server, Oracle RAC does not provide failover for database connections. In-flight transactions usually roll back when the database is the transaction manager. When the WebLogic server is the transaction manager, in-flight transactions are failed over in the sense that they are driven to completion or rolled back, based on the state of the transaction at the time of the failure.

# Oracle GridLink for RAC

- *GridLink* refers to the integration strategy between Fusion Middleware and RAC.
- Whenever possible, Oracle recommends using ***WLS multi data sources*** over the JDBC driver's load balancing and failover features.
- For Oracle DB 10g and earlier, XA is supported only with multi data sources.



**ORACLE**

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## WLS and RAC

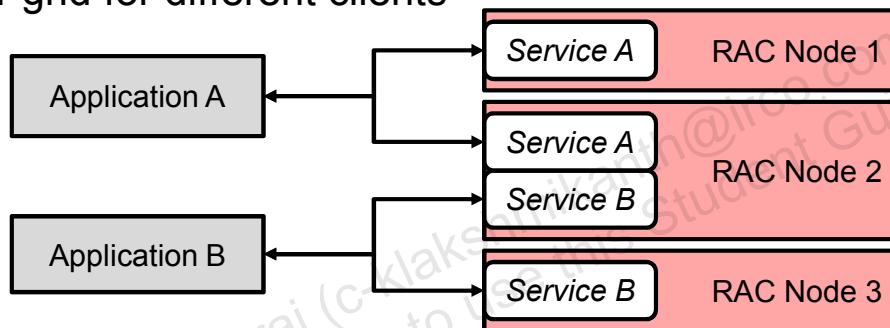
Because every RAC node in the cluster has equal access and authority, the loss of a node may impact performance but does not result in downtime. Depending upon your configuration, when a RAC node fails, in-flight transactions are redirected to another node in the cluster either by WebLogic Server or by the Oracle Thin driver. Note that Oracle RAC does not provide failover for database connections, nor does WebLogic Server. But transactions are failed over in the sense that they are driven to completion, based on the state of the transaction at the time of the failure.

For Oracle RAC deployment with Oracle Database 10g (10.2) and earlier, XA is supported only when using multi data sources. Therefore, load balancing for XA is supported only when using multi data sources. In a configuration without a multi data source, WebLogic Server relies on the connect-time failover feature provided by the Oracle Thin driver to work with Oracle RAC. However, the Oracle Thin driver cannot guarantee that a transaction is initiated and concluded on the same Oracle RAC instance when the driver is configured for load balancing. Oracle Database 11g includes support for Oracle 11g RAC as a high-availability database solution for WebLogic Server. However, in this release, WebLogic Server does not support new features in Oracle 11g RAC that support XA with load balancing. Instead, WebLogic Server will continue to use its well-proven integration architecture using multi data sources for XA with load balancing. If multi data sources are not an option, load balancing for XA is not recommended because of the significant performance detriment it causes.

# RAC Services

Oracle DB supports optional services that:

- Act as gateways to RAC nodes
- Are assigned a preferred list of nodes
- Automatically start on another node if the current one fails
- Are accessed by clients using the service name
- Allow you to control and prioritize the available capacity on your grid for different clients



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## RAC Services

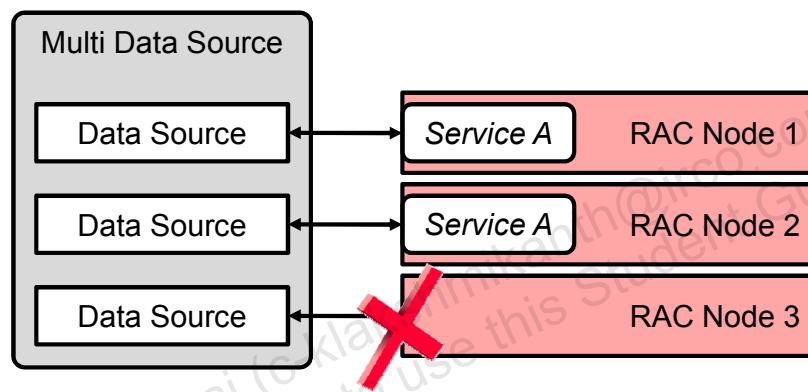
To manage workloads or a group of applications, you can define services that you assign to a particular application or to a subset of an application's operations. You can also group work by type under services. For example, online users can be a service, while batch processing can be another, and reporting can be yet another service type.

When you define a service, you define which instances normally support that service. These are known as the PREFERRED instances. You can also define other instances to support a service if the service's preferred instance fails. These are known as AVAILABLE instances. Oracle attempts to ensure that the service always runs on the number of instances for which you have configured the service. Afterwards, due to either instance failure or planned service relocations, a service may be running on an AVAILABLE instance. You cannot control which AVAILABLE instance to which Oracle relocates the services if there is more than one in the list. When a service moves to an AVAILABLE instance, Oracle does not move the service back to the PREFERRED instance when the PREFERRED instance restarts.

Distributed transaction processing applications have unique requirements. To make it easier to use Oracle RAC with global transactions, set the distributed transaction processing parameter on the service so that all tightly coupled branches of a distributed transaction processing transaction are run on the same instance.

## WLS and RAC Services

- Create a multi data source for each DB service.
- Configure member data source URLs with the service name and set their initial capacities to zero.
- Multi data sources will avoid nodes on which the service is not currently active (connections will fail).



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### WLS and RAC Services

If you rely on Oracle services in your RAC cluster for workload management, you must use multi data sources to connect to those services instead of you using a Service ID (SID). A WebLogic Server data source can be configured to connect only to a specific service on a specific RAC node, providing both workload management and load balancing.

In general, you need to create a multi data source for each service to which you want to connect. Within each multi data source, create one data source for each RAC node in the cluster on which the service will be configured, whether or not the service will be actively running on each node. Specify a different host name and port for each data source URL in a given multi data source as usual, but the service name URL parameter must be the same for all data sources.

It is recommended that you configure your data source's initial capacity to 0. This prevents pool creation failure for inactive pools at WLS startup, and enables WLS to create the data source even if it cannot connect to the service on the node. Without setting this option to 0, data source creation will fail and the server may fail to boot normally.

## **WLS and RAC Services (continued)**

For service connection scenarios, Oracle recommends that you configure your multi data source with the Load Balancing algorithm. If the multi data source is configured with the Load Balancing algorithm, its connection pools are used in a round robin fashion. In this case, workload is load-balanced across all of the RAC nodes on which the associated service is currently active.

If the multi data source is configured with the Failover algorithm, the first data source is used to connect to the service on its associated RAC node, until a connection attempt fails for any reason (for example, the RAC node becomes unavailable or there are no more connections available in the data source). At that point, the second data source is used to connect to the service on its associated RAC node, and so on. In this case, the RAC node to which the first data source is connected will experience more use than the remaining nodes on which the service is running.

In a workload management configuration, each multi data source has one data source configured for a given service on each RAC node, regardless of whether the service you are connecting to is active or inactive on a given RAC node. This lets you quickly start an inactive service on a node and create connections to that service should another node become unavailable due to unplanned downtime or scheduled maintenance. It also lets you quickly increase or decrease the available capacity for a given service based on workload demands.

When you start the service on a node, the associated data source detects that the service is now active, and the data source will then start making connections to that node as needed. When you stop a service on a given node, the associated data source can no longer make connections to that node, and will become inactive until the service is restarted on that node. The WLS data source performs connection testing. This lets the data source adjust to changes in the topology of the RAC configuration. The data source performs polling to see if its associated service is active or inactive. The connection test fails if the service is no longer available on the RAC node.

# Creating Data Sources to Support RAC Services

Service Name: **Service1** (circled with green circle labeled 2)

Database Name: **Node7**

Host Name: **myhost**

Database Type: **Oracle**

Port: **1521**

Database Driver: **\*Oracle's Driver (Thin XA) for RAC Service-Instance connections** Versions: 10,11

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Creating Data Sources to Support RAC Services

1. Select Data Sources and click New. For type, select Oracle and click Next. Then, for database driver, select the Oracle Driver (Thin XA) for RAC Service-Instance Connections option, and click Next. If your RAC setup does not use services, select the standard Oracle Driver (Thin XA) option instead.
2. Click Next again. Then, in addition to the database name and host name of the specific RAC node that you want to connect to, provide a service name.

The resulting JDBC URL for the above example will be similar to:

```
jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=myhost)(PORT=1521)))(CONNECT_DATA=(SERVICE_NAME=Service1)(INSTANCE_NAME=Node7))).
```

The SERVICE\_NAME parameter must be the same for all data sources in a given multi data source. But specify a different host and/or port for each data source in a given multi data source.

## Additional RAC Considerations

- Tune the test frequency of the multi data source based on the recovery time of a database instance.
- If you are using XA, tune the XA retry parameters of each data source to avoid lost transactions after database instance failure.
- If you are using XA and multi data source load balancing, all data sources should be configured so that a distributed transaction is directed to the same database instance.
- Failures while a connection is in use must still be handled by the application.



**ORACLE**

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### WLS and RAC Considerations

The Test Frequency Seconds multi data source attribute controls the frequency at which WebLogic Server checks the health of data sources previously marked as unhealthy to see whether connections can be re-created and if the data source can be enabled again. For fast failover of RAC nodes, set this value to a smaller interval such as 10 (seconds).

Occasionally, when one RAC node fails over to another, there may be a delay before the data associated with a transaction branch in progress on the now failed node is available throughout the cluster. This prevents incomplete transactions from being properly completed, which could further result in data locking in the database. To protect against the potential consequences of such a delay, WebLogic Server provides two data source configuration attributes that enable XA call retry for Oracle RAC: XA Retry Duration and XA Retry Interval.

XA Retry Duration controls the period of time during which WebLogic Server will repeatedly retry XA operations such as recover, commit, and roll back for pending transactions. XA Retry Interval controls the frequency of the retry attempts within the established time period.

Use the following formula to determine the value for XA Retry Duration: (longest transaction timeout for transactions that use connections from the data source) + (delay before Global Transaction IDs (XIDs) are available on all RAC nodes, typically less than five minutes).

## **WLS and RAC Considerations (continued)**

An additional data source attribute that pertains to Oracle RAC is Keep XA Connection Until Transaction Complete. This attribute forces the data source to reserve a physical database connection and provide the same connection to an application throughout transaction processing until the distributed transaction is complete. Confirm that this feature is enabled for proper transaction processing with Oracle RAC.

If you choose to specify them, all XA-related attributes must be set to the same values for each data source.

# Quiz

Which of the following is a data source attribute associated with connection testing?

- a. Test Statement
- b. Test Capacity
- c. Test Table Name
- d. Test LLR



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

**Answer: c**

# Quiz

Name three attributes used to configure a multi data source.

- a. Statement Cache Type
- b. Logging Last Resource
- c. Algorithm Type
- d. Failover Request if Busy
- e. JNDI Name



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

**Answer:** c, d, e

## Summary

In this lesson, you should have learned how to:

- Explain multi data source load balancing and failover algorithms
- Configure connection testing for a data source
- Define a new multi data source by using the console or WLST
- Describe a recommended approach to integrate WebLogic Server with Oracle RAC



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Practice 6-1: Use a Multi Data Source for Failover

This practice covers the following topics:

- Enabling data source connection testing
- Defining and targeting a multi data source
- Managing multi data source membership

# JDBC Performance Essentials

ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

# Objectives

After completing this lesson, you will be able to:

- Discuss common Java Database Connectivity (JDBC) application performance issues with your development team
- Perform some simple modifications to affect the performance of data sources and transactions
- Describe WebLogic Server's Logging Last Resource (LLR) feature

# JDBC and Application Design

Data source performance is most affected by application design and implementation:

- Cache a data source to avoid excessive Java Naming and Directory Interface (JNDI) lookups
- Reuse a connection within a single UI transaction
- Do not cache a connection across multiple UI transactions
- Avoid non-XA work within distributed transactions
- Use PreparedStatements and batching



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

ORACLE

## JDBC and Application Design

Most performance gains or losses in a database application are not determined by the application language, but by how the application is designed. The number and location of clients, size, and structure of database management system (DBMS) tables and indexes, and the number and types of queries all affect application performance.

Whenever possible, collect a set of data operations and submit an update transaction in one statement. This approach results in better performance than using separate statements and commits.

A simple way to boost JDBC application performance and avoid wasting resources:

- JNDI lookups are relatively expensive, so caching an object that required a looked-up in client code or application code avoids incurring this performance hit more than once.
- After the client or application code has a connection, maximize the reuse of this connection rather than closing and reacquiring a new connection. Although acquiring and returning an existing creation is much less expensive than creating a new one, excessive acquisitions and returns to pools create contention in the connection pool and degrades application performance.
- Do not hold connections any longer than is necessary to achieve the work needed. Getting a connection once, completing all necessary work, and returning it as soon as possible provides the best balance for overall performance.

# Connection Pooling

- Connection creation is expensive.
- For applications that consistently involve heavy database traffic:
  - Determine the optimal maximum capacity of a data source experimentally
  - Set the **Initial Capacity** and **Maximum Capacity** to the same value
- For applications whose peak database load is periodic or intermittent:
  - Use different values for initial and maximum sizes
  - Tune the **Capacity Increment** and **Shrink Frequency** based on the speed at which the load changes



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Connection Pool Capacity

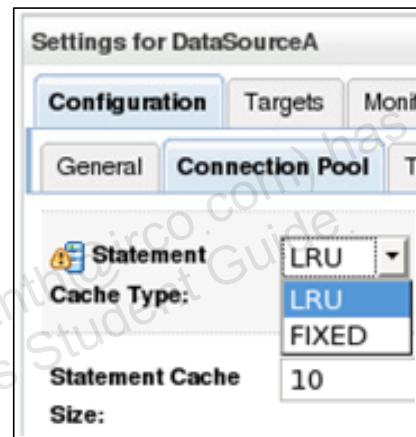
Each JDBC data source has a pool of JDBC connections that are created when the data source is deployed or at server startup. Applications use a connection from the pool and then return it when finished using the connection. Connection pooling enhances performance by eliminating the costly task of creating database connections for the application.

Creating a database connection is a relatively expensive process in any environment. Typically, a connection pool starts with a small number of connections. As client demand for more connections grows, there may not be enough in the pool to satisfy the requests. WebLogic Server creates additional connections and adds them to the pool until the maximum pool size is reached.

One way to avoid connection creation delays for clients using the server is to initialize all connections at server startup, rather than on-demand as clients need them. Set the initial number of connections equal to the maximum number of connections on the Connection Pool tab of your data source configuration. However, you will still need to determine the optimal value for the maximum capacity as part of your preproduction performance testing.

# Statement Caching

- JDBC PreparedStatements and CallableStatements require some additional overhead but improve overall performance through reuse.
- Data sources can automatically cache and reuse these statements:
  - For each connection
  - Across application requests
- By default, the *least recently used (LRU)* statements are removed from the cache when full.



**ORACLE**

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Statement Caching

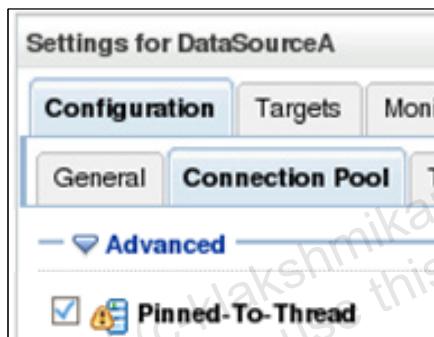
When you use a prepared statement or callable statement in an application or EJB, there is considerable processing overhead for the communication between the application server and the database server and on the database server itself. To minimize the processing costs, WebLogic Server can cache prepared and callable statements used in your applications. When an application or EJB calls any of the statements stored in the cache, WebLogic Server reuses the statement stored in the cache. Reusing prepared and callable statements reduces CPU usage on the database server, improving performance for the current statement and leaving CPU cycles for other tasks.

Each connection in a data source has its own individual cache of prepared and callable statements used on the connection. However, you configure statement cache options per data source. That is, the statement cache for each connection in a data source uses the statement cache options specified for the data source, but each connection caches its own statements.

If the statement is not in the cache, and the cache is full (the number of statements in the cache equals the statement cache size), WebLogic Server determines which existing statement in the cache was the least recently used and replaces that statement in the cache with the new statement.

## Connection Pinned to Thread

- A data source can dedicate or “pin” a connection to the first server thread that requests it.
- This capability:
  - May increase performance by eliminating potential contention for connections by threads
  - Is not supported with multi data sources or Oracle RAC



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Connection Pinned to Thread

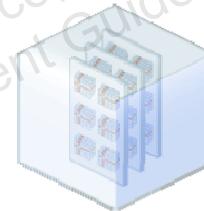
To minimize the time it takes for an application to reserve a database connection from a data source and to eliminate contention between threads for a database connection, you can enable the Pinned-To-Thread property in the connection properties.

When Pinned-To-Thread is enabled, WebLogic Server pins a database connection from the data source to an execution thread the first time an application uses the thread to reserve a connection. When the application finishes using the connection and calls `connection.close()`, which otherwise returns the connection to the data source, WebLogic Server keeps the connection with the execute thread and does not return it to the data source. When an application subsequently requests a connection using the same execute thread, WebLogic Server provides the connection already reserved by the thread. There is no locking contention on the data source that occurs when multiple threads attempt to reserve a connection at the same time and there is no contention for threads that attempt to reserve the same connection from a limited number of database connections.

In this release, the Pinned-To-Thread feature does not work with multi data sources or Oracle RAC. These features rely on the ability to return a connection to the connection pool and reacquire it if there is a connection failure or if a connection identity does not match.

## Logging Last Resource (LLR) Transactions

- LLR is a WebLogic optimization that allows one non-XA resource to participate in a distributed transaction:
  - Delays the non-XA data source until after all other XA work is prepared
  - Commits the XA transaction based on the outcome of the non-XA data source
  - Tends to yield performance gains for insert, update, and delete operations, but not read operations
- Additional tables are required for each server to track LLR transactions (`WL_LLRL_servername`).



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

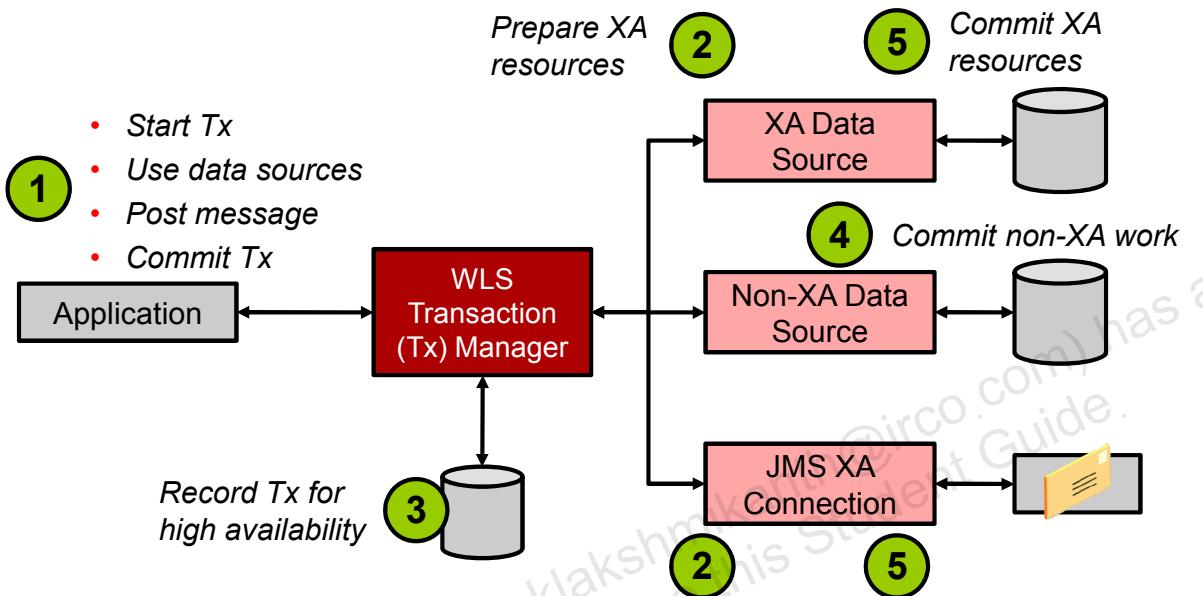
### Logging Last Resource (LLR) Transactions

WebLogic Server includes support for the Logging Last Resource (LLR) transaction optimization through JDBC data sources. LLR is a performance enhancement option that enables one non-XA resource to participate in a global transaction with the same atomicity, consistency, isolation, durability (ACID) guarantee as XA. The LLR resource uses a local transaction for its transaction work. The WebLogic Server transaction manager prepares all other resources in the transaction and then determines the commit decision for the global transaction based on the outcome of the LLR resource's local transaction.

In many cases a global transaction becomes a two-phase commit (2PC) transaction because it involves a database operation (using JDBC) and another nondatabase operation, such as a message queuing operation (using JMS). In cases such as this where there is one database participant in a 2PC transaction, the Logging Last Resource (LLR) Optimization transaction option can significantly improve transaction performance by eliminating some of the XA overhead for database processing and by avoiding the use of JDBC XA drivers, which typically are less efficient than non-XA drivers.

When an LLR transaction is committed, the WebLogic Server transaction manager handles the processing transparently. From an application perspective, the transaction semantics remain the same, but from an internal perspective, the transaction is handled differently than standard XA transactions.

## LLR Example



ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

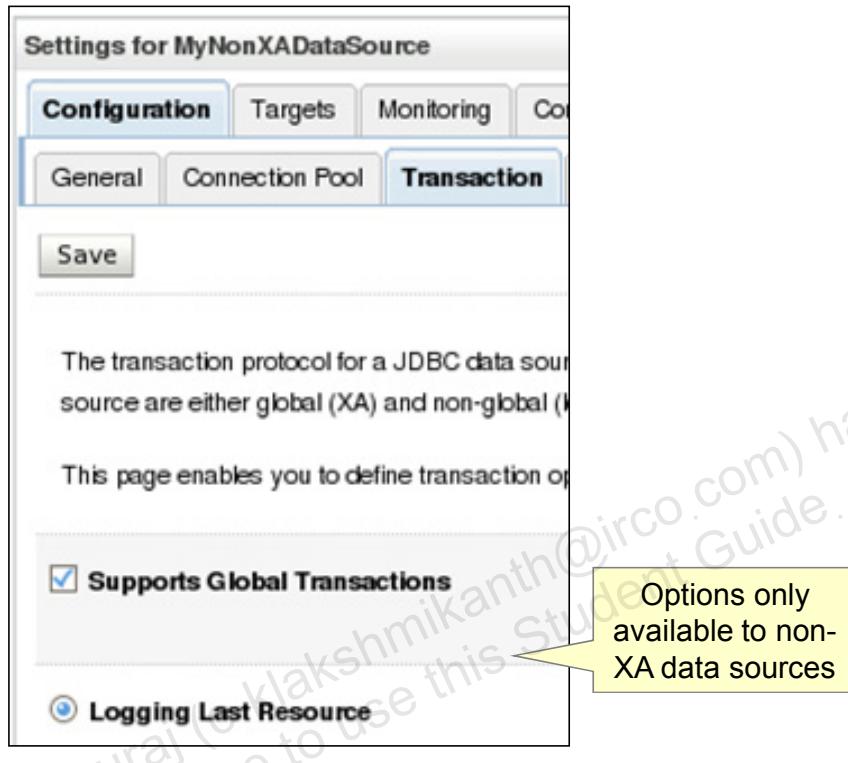
### LLR Example

Each WebLogic server instance maintains a database LLR table on the database to which a JDBC LLR data source pools database connections. These tables are used for storing transaction log records, and are automatically created. If multiple LLR data sources are deployed on the same WebLogic server instance and connect to the same database instance and database schema, they will also share the same LLR table. LLR table names are automatically generated unless administrators choose to configure them. The default table name is `WL_LLRSERVERNAME`.

In a global 2PC transaction with an LLR participant, the WebLogic Server transaction manager follows these basic steps:

1. Receives a commit request from the application
2. Calls a prepare on all other (XA-compliant) transaction participants
3. Inserts a commit record to a table on the LLR participant (rather than to the file-based transaction log)
4. Commits the LLR participant's local transaction (which includes both the transaction commit record insert and the application's SQL work)
5. Calls a commit on all other transaction participants
6. After the transaction completes successfully, lazily deletes the database transaction log entry as part of a future transaction

# Configuring LLR



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Configuring LLR

The LLR optimization improves performance by:

- Removing the need for an XA JDBC driver to connect to the database. XA JDBC drivers are typically inefficient compared to non-XA JDBC drivers.
- Reducing the number of processing steps to complete the transaction, which also reduces network traffic and the number of disk I/Os
- Removing the need for XA processing at the database level

The LLR optimization provides a significant increase in performance for insert, update, and delete operations. However, for read operations with LLR, performance is somewhat slower than read operations with XA. For best performance, you may want to configure a non-LLR JDBC data source for read-only operations.

The server will not boot if an LLR table is unreachable during boot. LLR transaction records must be available to correctly resolve in-doubt transactions during recovery, which runs automatically at server startup. If a transaction's coordinating server crashes before an LLR resource stores its transaction log record or before an LLR resource commits, the transaction rolls back. If the server crashes after the LLR resource is committed, the transaction will eventually fully commit. During server boot, the transaction coordinator will use the LLR resource to read the transaction log record from the database and then use the recovered information to commit any unfinished work on any participating non-LLR XA resources.

## Quiz

What data source attribute affects the growth of the connection pool size at run time?

- a. Capacity Increment
- b. Algorithm Type
- c. Cache Size
- d. Pinned To Thread



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

**Answer: a**

## Quiz

Name two values that the Statement Cache Type attribute supports?

- a. LRU
- b. Shrink
- c. Pinned
- d. Fixed



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

**Answer:** a, d

## Summary

In this lesson, you should have learned how to:

- List several administrative and programmatic techniques that can improve JDBC performance
- Compare and contrast strategies for tuning a data source connection pool

## JMS Message Management

8

ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

# Objectives

After completing this lesson, you will be able to:

- Inspect and manage in-progress Java Message Service (JMS) messages
- Create test JMS messages
- Pause JMS services for maintenance and troubleshooting

# WebLogic JMS Review

Resource	Description
Destination (queue or topic)	Applications use these to produce/consume messages; available on Java Naming and Directory Interface (JNDI)
Connection factory	Applications use these to connect to a JMS provider; each can be configured with different default communication settings; available on JNDI
JMS module	A group of related JMS resources, including destinations and factories, which is targeted to one or more servers
JMS server	Settings that configure how messages should be processed and persisted; targeted to a single server
Subdeployment	A reusable list of target JMS servers that can be associated with a resource in a JMS module



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## WebLogic JMS Review

A JMS destination identifies a queue (point-to-point) or topic (publish/subscribe) that can be used by applications to communicate asynchronously via messages.

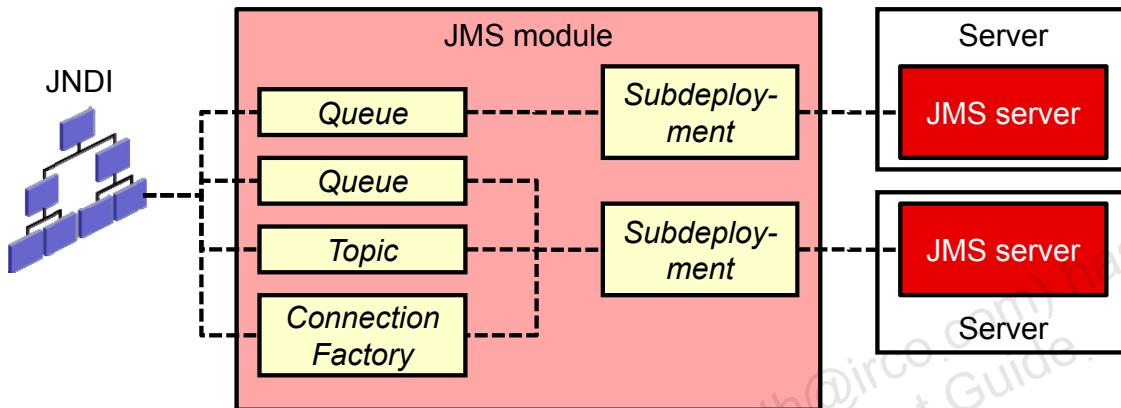
Connection factories are resources that enable JMS clients to create JMS connections. WebLogic JMS provides preconfigured default connection factories that can be enabled or disabled on a per-server basis. Otherwise, you can configure one or more connection factories to create connections with predefined options that better suit your application.

After you create a JMS module, you can configure resources for the module, including stand-alone queues and topics, distributed queues and topics, connection factories, and JMS templates.

A JMS server's primary responsibility for its destinations is to maintain information about the persistent store that is used for any persistent messages that arrive on the destinations and to maintain the states of the durable subscribers created on the destinations. JMS servers also manage message paging on destinations. Multiple JMS servers can be targeted to the same Oracle WebLogic Server instance.

While a JMS module is targeted to one or more servers, all of its resources need not be targeted homogeneously to these servers. Instead, modules can define subdeployments so that different resources are targeted to different server instances or JMS servers.

# WebLogic JMS Review



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## WebLogic JMS Review (continued)

WebLogic administrators typically use the administration console or the WebLogic Scripting Tool (WLST) to create and deploy (target) JMS modules and to configure the module's configuration resources, such as queues, and topics connection factories. JMS modules that you configure this way are considered system modules. JMS system modules are owned by the administrator, who can at any time add, modify, or delete resources. System modules are globally available for targeting to servers and clusters configured in the domain and therefore are available to all applications deployed on the same targets and to client applications.

JMS configuration resources can also be managed as deployable application modules, similar to standard Java EE descriptor-based modules. JMS application modules can be deployed either with a Java EE application as a packaged module, where the resources in the module are optionally made available to only the enclosing application (that is, application-scoped), or as a stand-alone module that provides global access to the resources defined in that module.

Unlike destinations, connection factories can be targeted to an entire WebLogic Server instance as well as a JMS server. In this case, the connection factory is available for use with destinations on all JMS servers on that server instance.

## Destination Management Features

- View or delete messages.
- Publish test messages.
- Move messages to another destination.
- Export message contents to a file.
- Import message contents from a file.
- Pause/resume message processing.



ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Destination Management Features

The WebLogic JMS message monitoring and management features allow you to create new messages, delete selected messages, move messages to another queue, export message contents to another file, import message contents from another file, or drain all the messages from the queue.

To perform these operations with WLST, access `JMSDestinationRuntimeMBean`. The available operations include:

- `deleteMessages`
- `getMessage`
- `importMessages`
- `moveMessages`
- `pause`
- `resume`

## Viewing Messages

	Name	Messages Current	Messages Pending	Messages Total
<input checked="" type="checkbox"/>	OrderJMSModule!OrderQueue	2	0	2

ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Viewing Messages

1. In the JMS Modules table, click the JMS module that contains the configured destination. In the selected JMS module's Summary of Resources table, click the queue or topic that you want to monitor or manage.
2. Click the Monitoring tab.
3. Select the check box next to the queue or topic.
4. Click Show Messages to access the queue's JMS Messages table.

## Viewing Messages

ID	State String	Time Stamp	Type	Message Size	Priority
ID:<64244.1238001924170.0>	visible	Wed Mar 25 13:25:24 EDT 2009	Text	83	2
ID:<64244.1238001924170.0>	visible	Wed Mar 25 13:25:24 EDT 2009	Text	83	2

ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Viewing Messages (continued)

- Click the ID link for a message in the queue to open the View Contents page, where you can view the contents of a JMS message. You can optionally filter the list of messages by using the Message Selector field (not shown). Message selectors are Boolean expressions based on message header or property values.

Click Delete to delete either all JMS messages from the current queue or to delete only those selected. The destination must be in a consumption-paused state, and the message state must be either visible, delayed, or ordered.

The available columns in the JMS Messages table include:

- State String:** The current state of a message
- Type:** The JMS message type, such as BytesMessage, TextMessage, StreamMessage, ObjectMessage, MapMessage, or XMLMessage
- Priority:** An indicator of the level of importance or urgency of a message, with 0 as the lowest priority and 9 as the highest
- Message Size:** The size of a message in bytes

# Message Contents

JMS Message Detail

<b>Message ID:</b>	ID:<64244.1238001924170.0:	<b>Delivery Mode:</b>	Persistent
<b>Type:</b>	Text	<b>Correlation ID:</b>	(No value)
<b>Timestamp:</b>	Wed Mar 25 13:25:24 EDT 2009	<b>Expiration:</b>	(No value)
<b>Priority:</b>	2	<b>Redelivered:</b>	false
<b>Delivery Time:</b>	(No value specified)	<b>Redelivery Limit:</b>	-1
<b>Text:</b>	<pre>&lt;order&gt;   &lt;id&gt;1234&lt;/id&gt;   &lt;user&gt;1234&lt;/user&gt;   &lt;total&gt;125.00&lt;/total&gt; &lt;/order&gt;</pre>		



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Message Contents

The JMS Message Detail page is a read-only view of a selected message's body (payload), headers, and properties. In addition to the fields described earlier, this page includes:

- **Delivery Mode:** The delivery mode is either persistent or nonpersistent.
- **Correlation ID:** A user-defined identifier for the message, often used to correlate messages about the same subject
- **Timestamp:** The time the message arrived on the destination
- **Expiration:** The expiration, or time-to-live value, for a message
- **Redelivered:** Indicates whether the message was redelivered or not
- **Redelivery Limit:** The number of redelivery tries a message can have before it is moved to an error destination
- **Properties:** Every JMS message contains a standard set of header fields that is included by default and available to all JMS providers and message consumers. Properties are other custom and provider-specific fields that can be set by message producers or the provider.

## Message States

A JMS message can be in one of several states in WebLogic Server, including:

- Receive
- Visible
- Send
- Transaction
- Delayed
- Ordered
- Expired
- Redelivery Count Exceeded
- Paused



ORACLE®

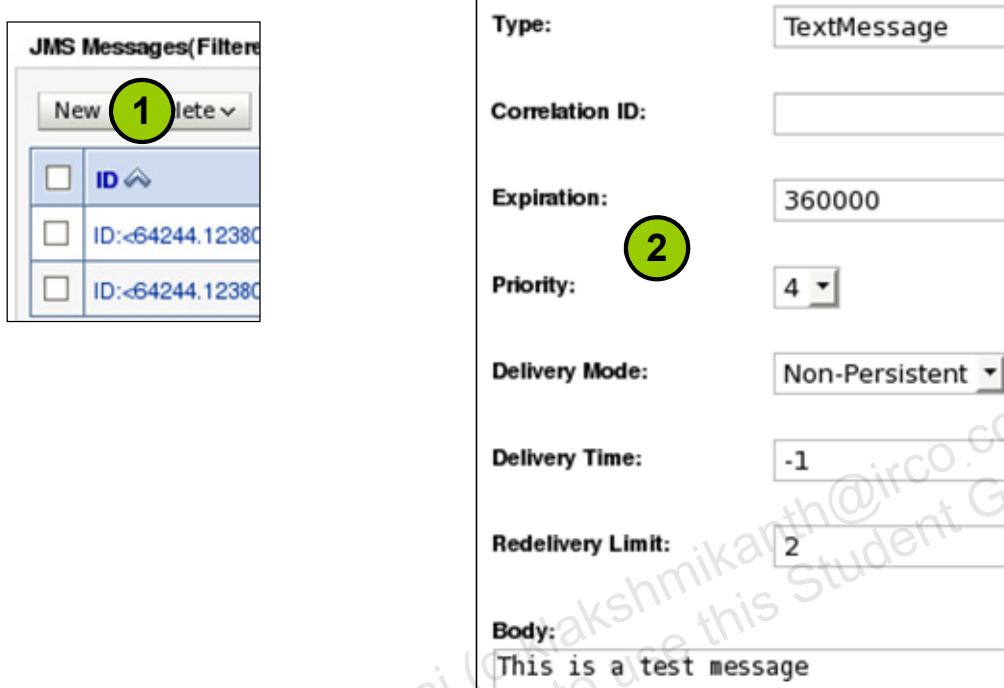
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Message States

As each JMS message is processed by WebLogic Server, it is assigned an internal state value. This state value changes as messages:

- Are initially received by a producer
- Are added to a destination and visible to consumers
- Are sent to consumers
- Participate in a transaction
- Time out based on their expiration setting
- Become unavailable to consumers because the destination was paused

## Publishing a Test Message



The image shows two windows side-by-side. On the left is a 'JMS Messages' table with three rows. The first row has a checkbox and the text 'ID'. The second row has a checkbox and the text 'ID:<64244.12380'. The third row has a checkbox and the text 'ID:<64244.12380'. A green circle with the number '1' highlights the 'New' button at the top of the table. On the right is a configuration dialog box with the following fields:

Type:	TextMessage
Correlation ID:	[empty]
Expiration:	360000
Priority:	4
Delivery Mode:	Non-Persistent
Delivery Time:	-1
Redelivery Limit:	2
Body:	This is a test message

ORACLE

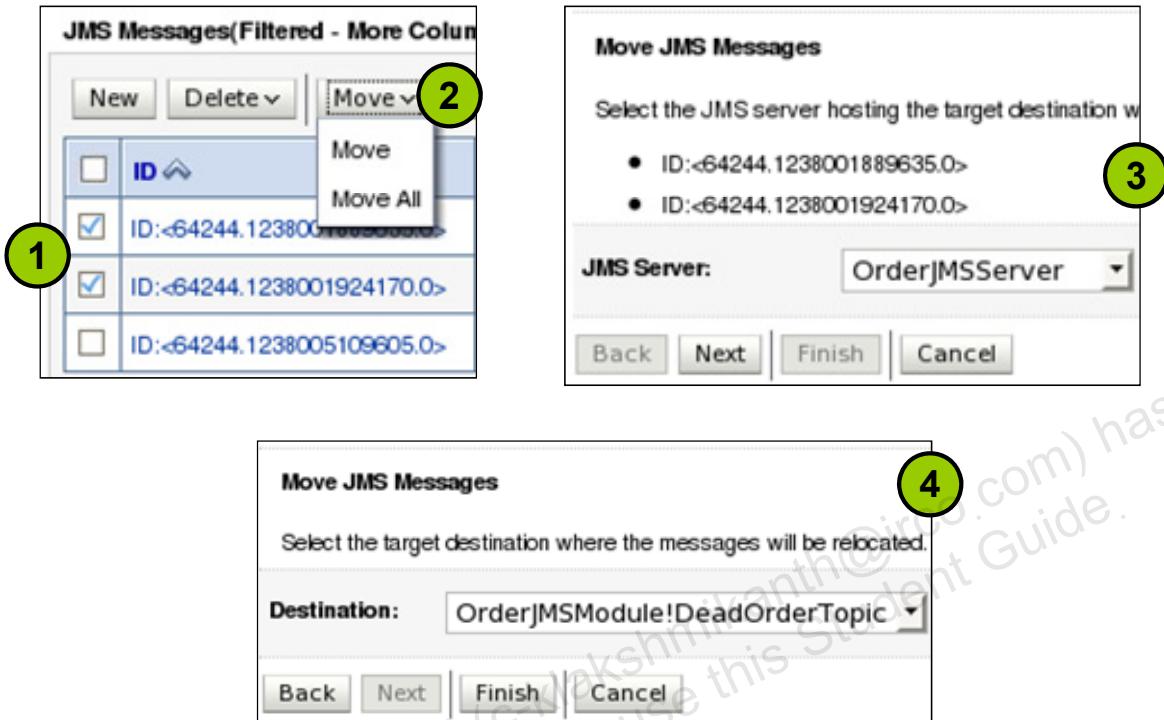
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Publishing a Test Message

To create a new message on a destination using the JMS Messages table:

1. Click the New button.
2. Provide the appropriate header values based on your specific test case, such as Type, Expiration, and Redelivery Limit. Enter the body of the message in the text box provided. When finished, click OK.

# Moving Messages



ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Moving Messages

Move JMS messages to another destination on either the current JMS server or on another JMS server in the cluster. The destination must be in a consumption-paused state, and the message state must be either “visible,” “delayed,” or “ordered.”

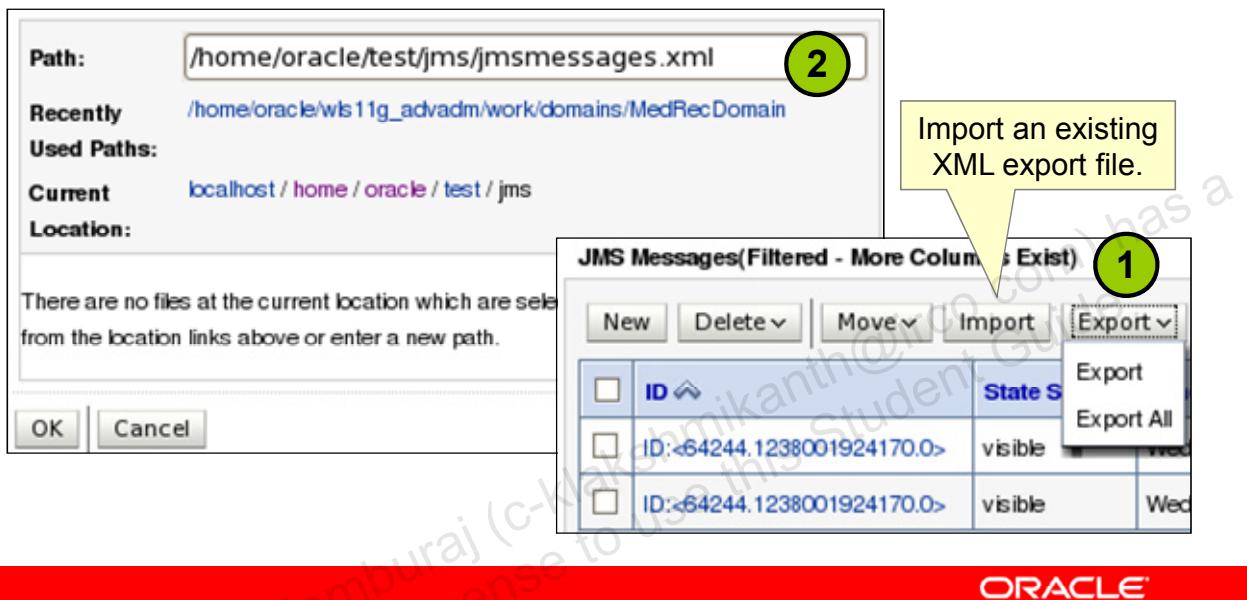
The message identifier does not change when you move a message. If the message being moved already exists on the target destination, a duplicate message with the same identifier is added to the destination.

To move messages on a destination using the JMS Messages table:

1. Select the check box next to the message(s) that you want to move
2. Click the Move list, and select Move or Move All
3. In the JMS Server field, select the JMS server hosting the target destination where the message(s) will be moved. Click Next.
4. In the Destination field, select the target destination where the message(s) will be moved. Click Finish.

# Exporting Messages to XML

To ensure data integrity, destinations should be paused before performing an export or import.



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

ORACLE

## Exporting Messages to XML

Exporting one or more JMS messages from a destination results in an XML file that can be imported into another destination on the same domain or a different domain. The source destination for an exported message can be either a queue, a distributed queue, or a topic durable subscriber. The source destination must also be in a production-paused state.

When a new message is created or an existing message is replaced with an imported file, the following rules apply:

- Quota limits are enforced for both new and replacement messages.
- The delivery count of the imported message is set to zero.
- A new message ID is generated for each imported message.
- If the imported message specifies a delivery mode of persistent and the target destination has no store, the delivery mode is changed to nonpersistent.

To export messages using the JMS Messages table:

1. Select the check box next to the message or messages that you want to move. Click the Export list, and select Export or Export All.
2. Use Path and the Current Location folders to select the target file to which the JMS messages will be exported. Click OK to export the file.

## Exported Messages Format

```
<?xml version="1.0" encoding="UTF-8"?>
<JMSSessageExport>

<mes:WLJMSMessage xmlns:mes="http://www.bea.com/WLS/JMS/Message">
  <mes:Header>
    <mes:JMSMessageID>ID:&lt;261352.1227725107824.0</mes:JMSMessageID>
    <mes:JMSDeliveryMode>PERSISTENT</mes:JMSDeliveryMode>
    <mes:JMSExpiration>0</mes:JMSExpiration>
    <mes:JMSPriority>4</mes:JMSPriority>
    <mes:JMSRedelivered>false</mes:JMSRedelivered>
    <mes:JMSTimestamp>1227725107824</mes:JMSTimestamp>
    <mes:Properties>
      <mes:property name="JMSXDeliveryCount">
        <mes:Int>0</mes:Int>
      </mes:property>
    </mes:Properties>
  </mes:Header>
  <mes:Body>
    <mes:Text>This is my first test message.</mes:Text>
  </mes:Body>
</mes:WLJMSMessage>
<mes:WLJMSMessage xmlns:mes="http://www.bea.com/WLS/JMS/Message">
  <mes:Header>
    <mes:JMSMessageID>ID:&lt;261352.1227725141980.0</mes:JMSMessageID>
    <mes:JMSDeliveryMode>PERSISTENT</mes:JMSDeliveryMode>
    <mes:JMSExpiration>0</mes:JMSExpiration>
    <mes:JMSPriority>4</mes:JMSPriority>
```



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Exported Messages Format

The example in the slide shows a typical XML file resulting from a JMS destination export. This file defines two JMS messages, and for each includes its headers and body.

# Pausing Destinations

- To aid in testing or troubleshooting, you can pause a destination:
  - When it first starts (server boot)
  - Dynamically while it is running
- Individual operations can be paused/resumed:

Operation	Description
<b>Production</b>	Messages received from producer and added to the destination
<b>Insertion</b>	Received messages that are added to a destination from being part of a committed transaction
<b>Consumption</b>	Messages sent to consumer and removed from the destination after acknowledgment



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Pausing Destinations

For troubleshooting purposes, you can temporarily pause all runtime message production, insertion, and consumption operations on a queue-by-queue basis. These options allow you to assert administrative control of JMS destinations in the event of an external resource failure. For example, by temporarily pausing message production and insertion on destinations, you can effectively drain all the existing messages for troubleshooting purposes, and then resume production and insertions after the issue has been resolved.

You can pause and resume the following types of JMS operations on destinations:

- **Production:** Messages produced when a producer creates and sends a new message to a destination
- **Insertion:** Messages inserted as a result of in-flight work completion, as when a message is available upon commitment of a transaction; for example, when a message scheduled to be made available after a delay is made available on a destination
- **Consumption:** Messages consumed when they are removed from the destination

When you pause a JMS destination for production, new and existing producers attached to that destination are unable to produce messages for that destination. A producer that attempts to send a message to a paused destination receives an `IllegalStateException` error that indicates that the destination is paused. When the destination resumes from the production pause state, producers can create and send messages.

# Pausing a Destination



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Pausing a Destination

1. In the JMS Modules table, click the JMS module that contains the configured destination. In the selected JMS module's Summary of Resources table, click the queue or topic that you want to monitor or manage.
2. Click the Control tab.
3. Select the check box for the queue to control.
4. Select one of the message control operations:
  - Click the Production button, and then click the Pause option to stop the production of new messages on this queue. To resume message production, click the Production button, and then click the Resume option.
  - Click the Consumption button, and then click the Pause option to stop message consumption on this queue. To resume the consumption of messages, click the Consumption button, and then click the Resume option.
  - Click the Insertion button, and then click the Pause option to stop the insertion of "in-flight" messages on this queue. To resume the insertion of in-flight messages, click the Insertion button, and then click the Resume option.

# JMS Management WLST Examples

Delete all messages from a destination:

```
domainRuntime()  
queue = getMBean('/ServerRuntimes/serverA/JMSRuntime/  
    serverA.jms/JMSServers/myJMSServer/Destinations/  
    OrderJMSModule!OrderQueue')  
queue.deleteMessages('')
```

Pause a destination:

```
domainRuntime()  
queue = getMBean('/ServerRuntimes/serverA/JMSRuntime/  
    serverA.jms/JMSServers/myJMSServer/Destinations/  
    OrderJMSModule!OrderQueue')  
queue.pauseConsumption()
```



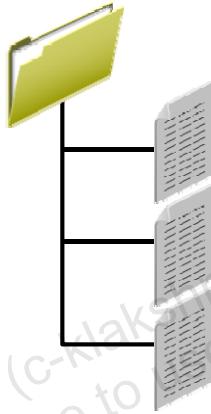
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## JMS Management WLST Examples

Refer to `JMSDestinationRuntimeMBean` in the documentation.

# JMS Logging

- Each JMS server generates its own log file, which:
  - By default is placed in the log folder for its target server and is named `jms.messages.log`
  - Supports rotation, similar to server logs
- Configure individual JMS destinations to log message state changes, along with selected message headers and properties.



ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## JMS Logging

After you create a JMS server, you can configure criteria for moving (rotating) old log messages to a separate file. You can also change the default name of the log file. You can choose to rotate old log messages to a new file based on a specific file size or at specified intervals of time. After selecting the JMS server in the console, click the Logging tab.

Rotated log files are numbered in the order of creation. For example, the seventh rotated file would be named `myserver.log00007`. For troubleshooting purposes, it may be useful to change the name of the log file or to include the time and date when the log file is rotated. To do this, you add `java.text.SimpleDateFormat` variables to the file name. Surround each variable with percentage (%) characters. If you specify a relative pathname when you change the name of the log file, it is interpreted as relative to the server's root directory.

When you enable message logging for a destination, you can specify whether the log entry will include all the message header fields or a subset of them, all system-defined message properties or a subset of them, or all user-defined properties or a subset of them. You may also choose to include or to exclude the body of the message.

# Configuring Destination Logging



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Configuring Destination Logging

After you create a queue, you can enable the logging of message life cycle information into a JMS message log file. A message life cycle is an external view of the basic events that a JMS message traverses through, such as message production, consumption, and removal. The content of the message log always includes message ID and correlation ID, but you can also configure information like message type and user properties.

To configure destination logging:

1. In the JMS Modules table, click the JMS module that contains the configured destination. In the selected JMS module's Summary of Resources table, click the queue or topic that you want to configure.
2. Click the Configuration > Logging tab.
3. Select the Enable Message Logging check box.
4. Select All Headers to include all JMS header fields in the log, or alternatively use Headers to include only a subset of JMS header fields in the log by moving them from the Available to the Chosen column.

## Quiz

Name four JMS management operations available in WLS.

- a. View the contents of a message
- b. Encrypt the contents of a message
- c. Add a message to a destination
- d. Pause the production of messages of a destination
- e. Move a message to another destination

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

**Answer:** a, c, d, e

## Quiz

Name three types of operations that can be individually paused on a JMS destination.

- a. Insertion
- b. Consumption
- c. Presumption
- d. Transaction
- e. Production



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

**Answer:** a, b, e

## Summary

In this lesson, you should have learned how to:

- View messages on a destination
- Create and delete messages on a destination
- Move messages within or across domains
- Pause a destination
- Enable and configure message logging

Unauthorized reproduction or distribution prohibited. Copyright© 2011, Oracle and/or its affiliates.

LakshmiKanth Kemburaj (c-klakshmikanth@irc0.com) has a  
non-transferable license to use this Student Guide.

## JMS Guaranteed Messaging

9

ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

# Objectives

After completing this lesson, you will be able to:

- Configure file and Java Database Connectivity (JDBC)-based Java Message Service (JMS) persistence
- Assign a persistent store to a JMS server
- View store contents using WebLogic Scripting Tool (WLST)
- Describe the concept of durable subscribership
- Manage and monitor durable subscriptions
- Explain how message paging works

## Guaranteed Messaging Concepts

WebLogic Server JMS includes several capabilities to help ensure that messages get delivered despite server or consumer failures, including:

- Persistent storage of in-progress messages
- Durable subscribership for topic consumers
- Paging of in-progress messages to preserve memory
- Automatic service migration in a cluster



ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Guaranteed Messaging Concepts

WebLogic JMS includes the following reliability features:

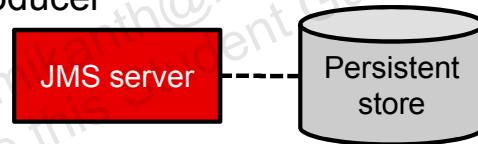
- JMS servers can use file or database persistent message storage (both fully XA transaction capable).
- For durable subscriptions, WebLogic JMS stores a message in a persistent file or database until the message has been delivered to the subscribers or has expired, even if those subscribers are not active at the time that the message is delivered.
- Transactions, including distributed transactions, between JMS applications and other transaction-capable resources are fully supported.
- If a server or network failure occurs, JMS producer and consumer objects will attempt to transparently fail over to another server instance in a cluster.
- Automatic whole server migration provides improved cluster reliability and server migration. WebLogic Server now supports automatic and manual migration of a clustered server instance and all the services it hosts from one machine to another.
- Failed or expired messages can be redirected to error destinations.

## Guaranteed Messaging Concepts (continued)

- The JMS Delivery Count message property specifies the number of message delivery attempts, where the first attempt is 1, the second is 2, and so on. WebLogic Server makes a best effort to persist the delivery count, so that the delivery count does not reset back to one after a server reboot.

# Persistent Messaging

- Each JMS Server can be assigned a persistent store, in which unconsumed messages are written to either of the following:
  - A file system
  - A JDBC database (requires a data source)
- Only messages marked as persistent will be written to the store:
  - Default connection factory settings
  - Default destination settings
  - Programmatically set by the producer



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Persistent Messaging

WebLogic persistent stores provide built-in, high-performance storage solutions for the Oracle WebLogic Server subsystems and services that require persistence. For example, they can store persistent JMS messages or temporarily store messages that are sent using the JMS store-and-forward feature. The persistent store supports persistence to a file-based store or to a JDBC-enabled database. Throughput is the main performance goal of the persistent store. Multiple subsystems can share the same persistent store, as long as they are all targeted to the same server instance or migratable target.

A persistent message is guaranteed to be delivered only once. The message cannot be lost due to a JMS provider failure and it must not be delivered twice. It is not considered sent until it has been safely written to a WebLogic persistent store that is assigned to each JMS server during configuration.

Nonpersistent messages are not stored, unless the separate paging feature has been enabled. They are guaranteed to be delivered at most once, unless there is a JMS provider failure, in which case the messages may be lost. If, however, a connection is closed or recovered, all nonpersistent messages that have not yet been acknowledged will be redelivered as normal.

## Default Persistent Store

- By default, a JMS server uses its target server's default file system store.
- The default store is:
  - Found at <server>\data\store\default
  - Configurable from the server's Configuration > Services tab
- Create custom stores to:
  - Use a database instead of a file system
  - Maintain different stores for different server subsystems
  - Support JMS and other service migration features within a cluster



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Default Persistent Store

Each server instance, including the administration server, has a default persistent store that requires no configuration. The default store is a file-based store that maintains its data in a group of files in a server instance's data\store\default directory. A directory for the default store is automatically created if one does not already exist. This default store is available to subsystems that do not require explicit selection of a particular store and function best by using the system's default storage mechanism. For example, a JMS server with no persistent store configured will use the default store for its managed server and will support persistent messaging.

You can specify another location for the default store using the console or WLST. In the left pane of the console, expand Environment and select Servers. Then select a server and click its Configuration > Services tab. Locate and edit the Directory field.

In addition to using the default file store, you can also configure a custom file store or JDBC store to suit your specific needs. One exception, however, is the server's transaction log (TLOG), which always uses the default store. This is because the TLOG must always be available early in the server boot process.

## Comparing File and JDBC Stores

- Both types guarantee persistence and support the same transaction semantics.
- Local file stores generate no network traffic, but a JDBC store typically does as it is on a separate machine.
- On similar hardware, a remote file store typically performs better than a JDBC store.
- JDBC stores support multi data sources and connection testing for failover.
- Network files stores tend to be easier to configure and manage than databases.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Comparing File and JDBC Stores

The following are some similarities and differences between file stores and JDBC stores:

- The default persistent store can only be a file store. Therefore, a JDBC store cannot be used as a default persistent store.
- Both have the same transaction semantics and guarantees. As with JDBC store writes, file store writes are guaranteed to be persisted to disk and are not simply left in an intermediate (that is, unsafe) cache.
- All things being equal, file stores generally offer better throughput than a JDBC store. However, you should always determine this experimentally, because performance can vary depending on your specific OS, hardware, and network configuration. For example, if a database is running on high-end hardware with very fast disks, and WebLogic Server is running on slower hardware or with slower disks, you may get better performance from the JDBC store.
- File stores are generally easier to configure and administer and do not require that WebLogic subsystems depend on any external component.
- JDBC stores may make it easier to handle failure recovery because the JDBC interface can access the database from any machine on the same network. With the file store, the disk must be shared or migrated.

# Creating a File Store



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

ORACLE

## Creating a File Store

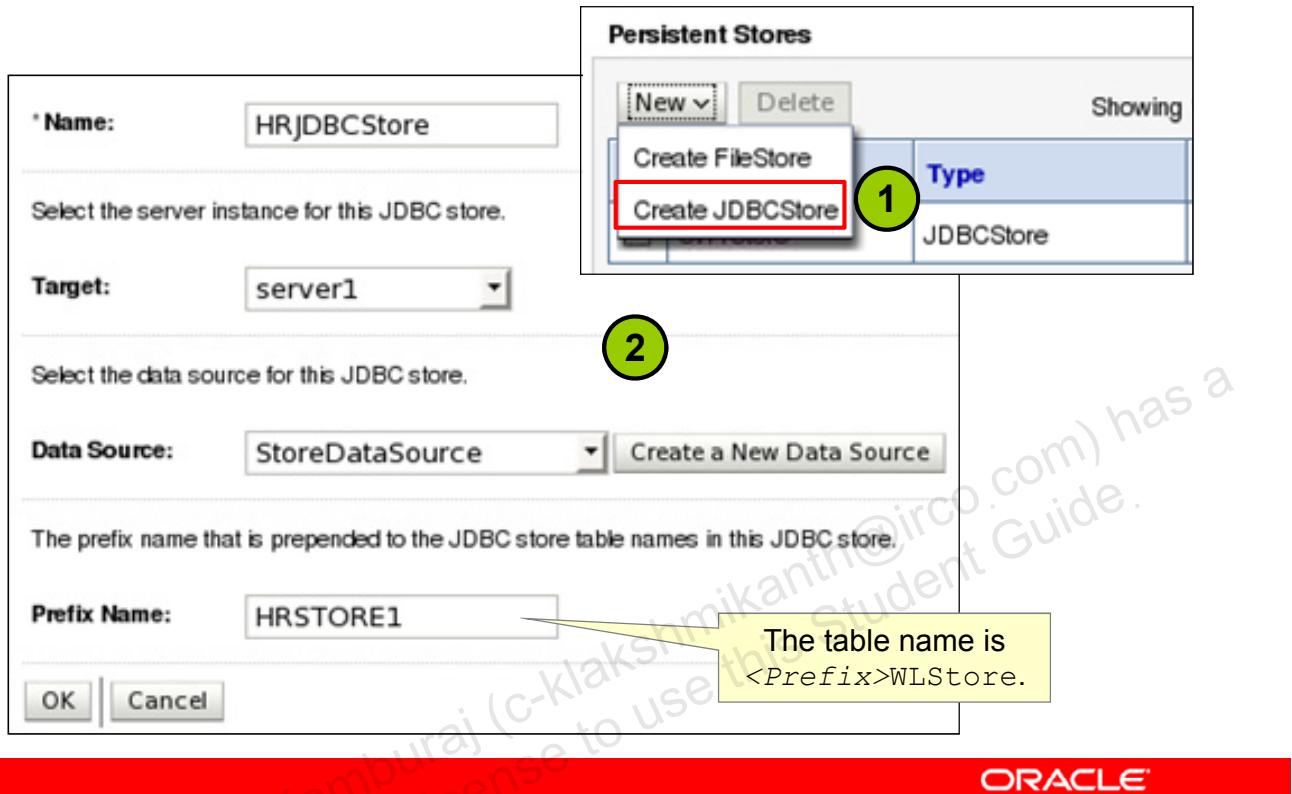
When configuring a file store directory, the directory must be accessible to the server instance on which the file store is located. Perform the following steps:

1. In the left pane of the console, expand Services, and select Persistent Stores.
2. On the Summary of Persistent Stores page, select New > Create FileStore.
3. Update the following:
  - **Name:** Name of the store
  - **Target:** Server instance on which to deploy the store
  - **Directory:** Pathname to the directory on the file system where the file store is placed. This directory must exist on your system, so be sure to create it before completing this tab. For highest availability, use either a Storage Area Network (SAN) or a dual-ported SCSI disk.

If you edit an existing file store, you can also modify its Synchronous Write Policy field, which defines how hard a file store will try to flush records to the disk. The available values are Direct-Write (default), Cache-Flush, and Disabled.

To properly secure file store data, you must set appropriate directory permissions on all your file store directories. If you require data encryption, you must use appropriate third-party encryption software.

## Creating a JDBC Store



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

ORACLE

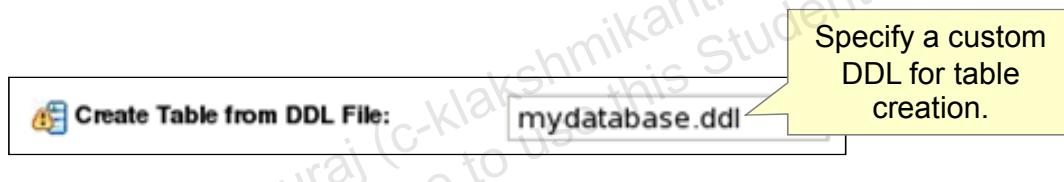
### Creating a JDBC Store

Perform the following steps:

1. On the Summary of Persistent Stores page, select New > Create JDBCStore.
2. Update the following:
  - **Name:** Name of the store
  - **Target:** Server instance on which to deploy the store
  - **Data Source:** Select a configured JDBC data source or multi data source used to access the JDBC store. If you need to configure a JDBC data source for the store, click the Configure a Data Source button. This data source or multi data source must be targeted to the same server instance as the JDBC store.
  - **Prefix Name:** When using multiple JDBC stores, you must set this option to a unique value for each configured JDBC store. When no prefix is specified, the JDBC store table name is simply WLStore, and the database implicitly determines the schema according the current user of the JDBC connection.

## JDBC Store Details

- Store data sources must use a non-XA driver.
- Multiple stores cannot share the same table.
- WLS will automatically attempt to create the required table if:
  - The table is not already present
  - The data source credentials have “Create” rights
  - The data source vendor type is not “Other”
  - A matching DDL file is found for the data source type
- Store DDL files for standard vendors are found within the archive:  
`<WL_HOME>/modules/com.bea.core.store.jdbc_xxx.jar.`



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

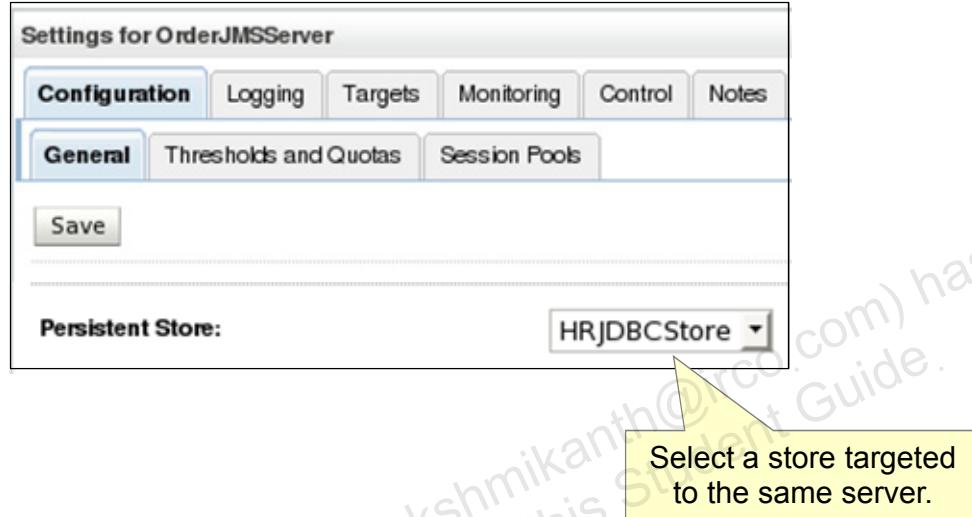
### JDBC Store Details

You cannot configure a JDBC store to use a JDBC data source that is configured to support global transactions. Instead, it must use a JDBC data source that uses a non-XA JDBC driver. You also cannot enable Logging Last Resource or Emulate Two-Phase Commit in the data source. This limitation does not remove the XA capabilities of layered subsystems that use JDBC stores. For example, WebLogic JMS is fully XA-capable regardless of whether it uses a file store or any JDBC store.

When configuring a data source, WebLogic Server detects some drivers for supported databases. For each of these databases, there are corresponding data definition language (DDL) files within the `WL_HOME\modules\com.bea.core.store.jdbc_xxx.jar` file, where `WL_HOME` is the top-level installation directory of your WebLogic Server installation. Within this archive, the files are found at `weblogic/store/io/jdbc/ddl`.

For Oracle databases, the default DDL used is `oracle.ddl`. However, you can instead override this file using the store’s “Create Table from DDL File” field. For example, an alternative file named `oracle_blob.ddl` is available that uses a BLOB data type instead of the default LONG RAW type.

## Assigning a Store to a JMS Server



ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Assigning a Store to a JMS Server

Associate your new custom persistent store with a JMS server by using the Persistent Store field of the JMS server's Configuration > General tab. If this field is set to "none," the JMS server uses the default file store that is automatically configured on the target server instance.

## Persistent Store WLST Example

Create a new JDBC store and assign it to a JMS server:

```
edit()
startEdit()

jdbcStore = create('HRJDBCStore', 'JDBCStore')
jdbcStore.setPrefixName('HRStore1')
jdbcStore.setDataSource(getMBean('/SystemResources/
    StoreDataSource'))
jdbcStore.addTarget(getMBean('/Servers/server1'))

jmsServer = getMBean('/JMServers/HRJMServer')
jmsServer.setPersistentStore(jdbcStore)

save()
activate(block='true')
```



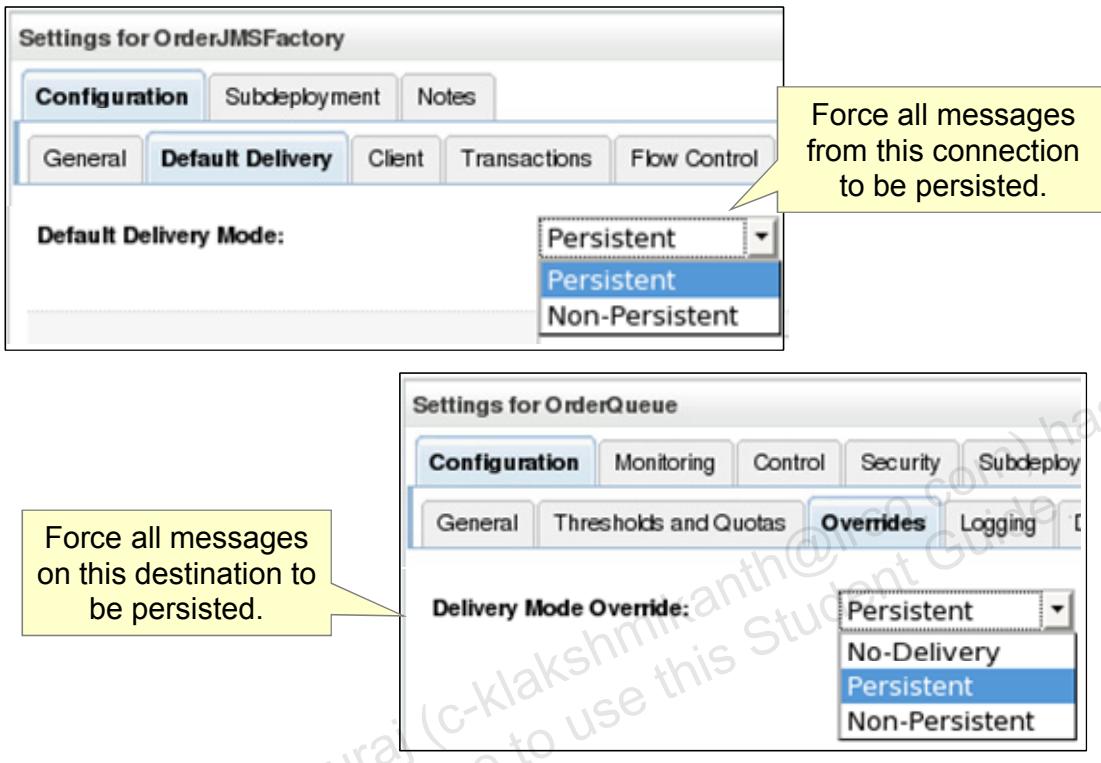
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Persistent Store WLST Example

Refer to the documentation on the following MBeans:

- DefaultFileStoreMBean
- FileStoreMBean (subtype of PersistentStoreMBean)
- JDBCStoreMBean (subtype of PersistentStoreMBean)
- JMServerMBean

# Connection Factory and Destination Persistence



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

ORACLE

## Connection Factory and Destination Persistence

After you configure a connection factory, you can define various default message delivery parameters that also override any values configured by the client. For example, if a client does not specify certain delivery parameters then the value of those parameters can be controlled with the delivery parameters on this page:

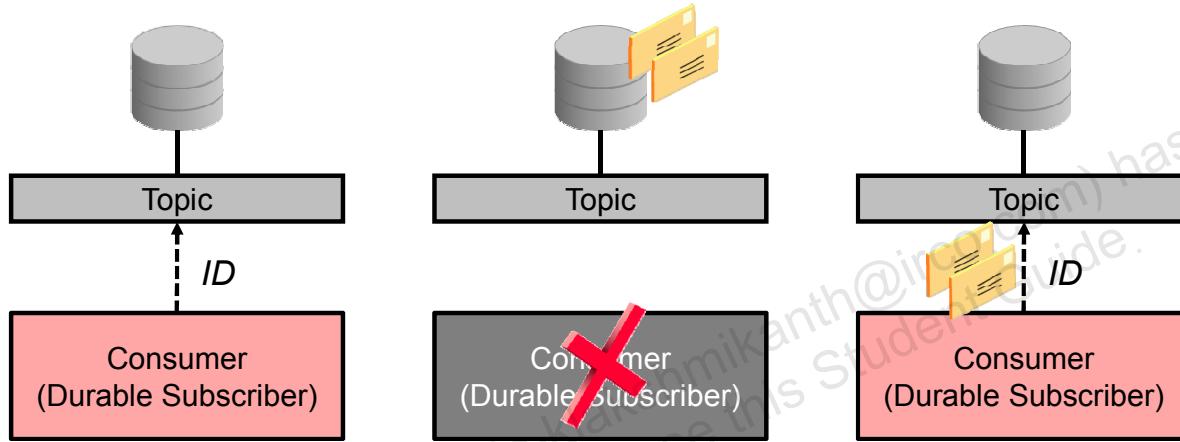
1. Select the connection factory.
2. Click Configuration > Default Delivery.
3. In Default Delivery Mode, select the delivery mode (Persistent or Non-Persistent) assigned to all messages sent by a producer using this connection factory.

After you create a queue or topic, you can define message delivery override values that can override those specified by a message producer.

1. Select the queue or topic.
2. Click Configuration > Overrides.
3. In Delivery Mode Override, select the delivery mode (Persistent, Non-Persistent, or No-Delivery) assigned to all messages that arrive at the queue. A value of No-Delivery specifies that the producer's delivery mode will not be overridden. This attribute is dynamically configurable, but only incoming messages are impacted; stored messages are not impacted.

## Topics and Durable Subscribers

- By default, inactive or failed topic consumers will miss any messages produced in their absence.
- If consumers register as durable and provide a unique client ID, messages will be saved across connections.



ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Topics and Durable Subscribers

Support for durable subscriptions is a feature that is unique to the publish and subscribe messaging model, so client IDs are used only with topic connections. Queue connections also contain client IDs, but JMS does not use them.

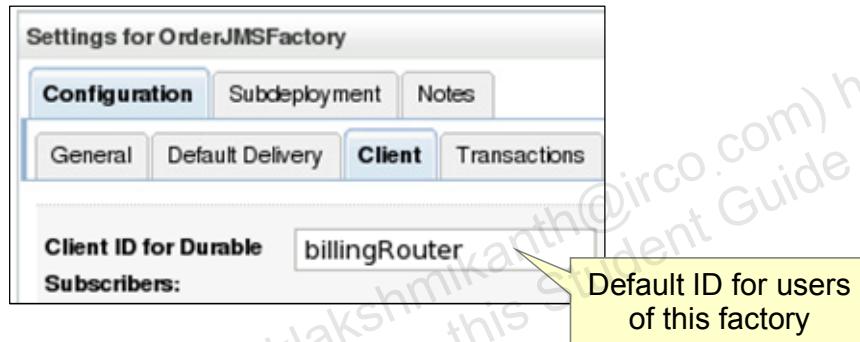
Nondurable subscriptions only last for the lifetime of their current session with the JMS server. That is, a client will only see the messages published on a topic while it is active. If the subscriber is not active, it is potentially missing messages that are published on its topic. By default, subscribers are nondurable.

A durable subscriber, on the other hand, registers a durable subscription with a unique identity that is retained by JMS. Subsequent subscriber objects with the same identity resume the subscription in the state it was left in by the previous subscriber. If there is no active subscriber for a durable subscription, JMS retains the subscriber's messages until they are received by the subscriber or until they expire.

## Connection Factory Client ID

Durable subscribers are associated with a client ID:

- Programmatically
- Using the default ID of the connection factory (create dedicated factories for each consumer)



**ORACLE**

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Connection Factory Client ID

The client ID can be supplied in two ways. The first method is to configure the connection factory with the client ID using the Configuration > Client tab in the console. This approach means adding a separate connection factory definition during configuration for each client ID. Applications then look up their own topic connection factories in Java Naming and Directory Interface (JNDI) and use them to create connections containing their own client IDs.

Alternatively, the preferred method is for an application to programmatically set its client ID in the connection after the connection is created. If you use this alternative approach, you can use the default connection factory (if it is acceptable for your application) and avoid the need to modify the configuration information.

Durable subscriptions should not be created for a temporary topic, because a temporary topic is designed to exist only for the duration of the current connection.

An application can use a JMS connection to both publish and subscribe to the same topic. Because topic messages are delivered to all subscribers, an application can receive messages it has published itself. To prevent this, a JMS application or a connection factory can set a No Local attribute to the value true. The default value is false.

# Monitoring Durable Subscribers

Settings for OrderJMSModule!OrderTopic

Configuration Monitoring Control Security Subdeployment Notes

Statistics Durable Subscribers

Durable Subscribers(Filtered - More Columns Exist)

New Delete Show Messages Showing 1 to 1 of 1 Previous | Next

	Subscription Name	Client ID	Active	Messages Current Count
<input checked="" type="checkbox"/>	BillingRouterApp	billingRouter	false	0

Show saved messages.

Manually create a subscription.

Is subscriber connected?

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Monitoring Durable Subscribers

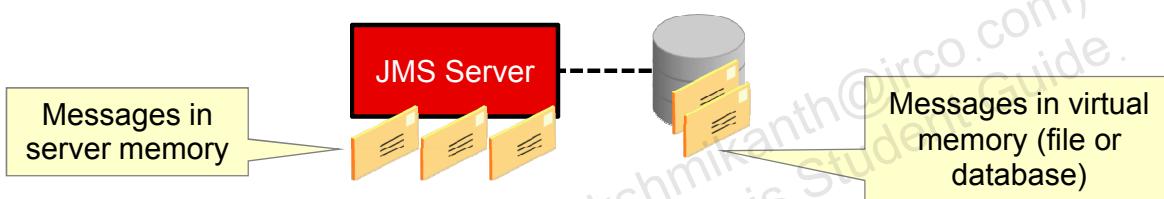
1. Select a topic in a JMS module.
2. Click Monitoring > Durable Subscribers to display the topic's subscription management options.
3. In addition to monitoring statistics for each subscriber, you can perform one of the following:
  - Click New to create a new durable subscriber.
  - Click Delete to delete specific durable subscribers.
  - To manage the messages for a durable subscriber, select the check box next to the durable subscriber's name, and then click Show Messages to access its message management page.

The available data columns include:

- **Client ID:** A unique client identifier for this durable subscriber
- **No Local Messages:** Specifies whether this durable subscriber receives local messages it has published itself
- **Active:** Indicates whether this subscription is being used by a durable subscriber
- **Messages Current Count:** The number of messages still available for this durable subscriber to consume

# Message Paging

- To avoid running out of memory under peak loads, JMS servers move in-progress messages to virtual memory (`servers/<server>/tmp`, by default).
- If using a persistent store, persistent messages are copied to both the store and the paging directory.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Message Paging

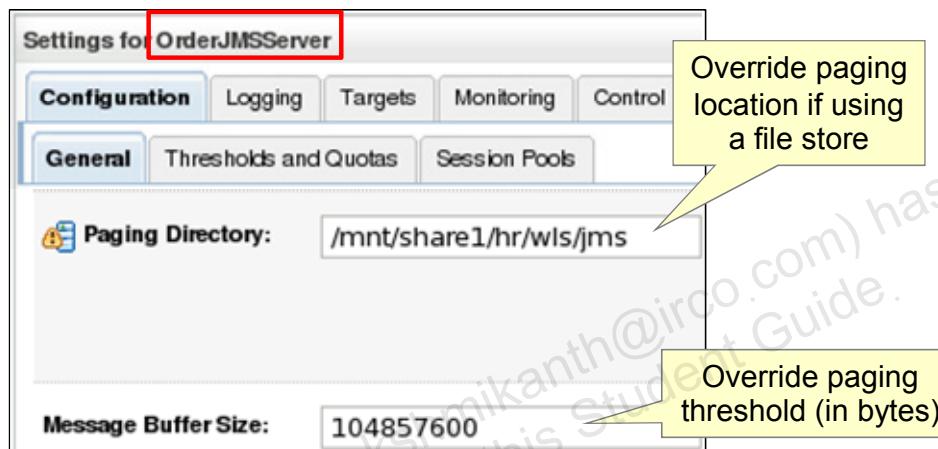
With the message paging feature, JMS servers automatically attempt to free up virtual memory during peak message load periods. This feature can greatly benefit applications with large message spaces. Message paging is always enabled on JMS servers, and so a message paging directory is automatically created without having to configure one. You can, however, specify a different directory if you want.

In addition to the paging directory, a JMS server uses either a file store or a JDBC store for persistent message storage. The file store can be user-defined or the server's default store. Paged persistent messages are copied to both the persistent store as well as the JMS server's paging directory. However, when using a file store, larger messages are not copied into the paging directory, only the store.

However, a paged-out message does not free all the memory that it consumes, because the message header, with the exception of any user properties that are paged out along with the message body, remains in memory for use with searching, sorting, and filtering. Queuing applications that use selectors to select paged messages may show severely degraded performance, because the paged out messages must be paged back in.

# Configuring Message Paging

By default, paging is triggered when messages consume more than one-third of maximum server memory (heap).



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Configuring Message Paging

If a paging directory is not specified for a JMS server, paged-out message bodies are written to the default \tmp directory inside the host server's root directory (<domain>/servers/<server>).

The Message Buffer Size JMS server attribute specifies the amount of memory that will be used to store message bodies in memory before they are paged out to disk. The default value is approximately one-third of the maximum heap size for the JVM. The larger this parameter is set, the more memory JMS will consume when many messages are waiting on queues or topics. After this threshold is crossed, JMS may write message bodies to the directory specified by the Paging Directory option in an effort to reduce memory usage below this threshold.

It is important to remember that this parameter is not a quota. If the number of messages on the server passes the threshold, the server writes the messages to disk and evicts the messages from memory as fast as it can to reduce memory usage, but it will not stop accepting new messages. It is still possible to run out of memory if messages are arriving faster than they can be paged out. Users with high messaging loads who want to support the highest possible availability should consider setting a quota, or setting a threshold and enabling flow control to reduce memory usage on the server.

## Quiz

Which of the following is true about JDBC stores?

- a. The store does not require a separate data source.
- b. Store DDL files must be placed in your domain.
- c. Each store must use a dedicated table.
- d. JMS connection factories must specify a client ID.

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

**Answer: c**

## Quiz

Which two values can the Delivery Mode message header be set to?

- a. Non-persistent
- b. At-most-once
- c. Exactly-once
- d. Persistent

 ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

**Answer:** a, d

## Summary

In this lesson, you should have learned how to:

- Configure file and JDBC stores for a server
- Assign a persistent store to a JMS server
- Configure JMS resource persistence settings
- Manage and monitor durable subscriptions for topics

## Practice 9-1: Configure JMS Persistence

This practice covers the following topics:

- Creating a new JDBC persistent store for a server
- Associating a JMS server with a persistent store
- Configuring persistence settings for JMS resources
- Verifying message persistence across server restarts



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

# 10

## JMS Performance Essentials

ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

# Objectives

After completing this lesson, you will be able to:

- List various Java Message Service (JMS) parameters that can affect performance
- Create quotas and thresholds for JMS servers or destinations
- Enable message compression
- Explain the purpose of JMS Unit of Order

## JMS and Application Design

JMS performance is most affected by application design and implementation:

- Cache factories and destinations to avoid excessive Java Naming and Directory Interface (JNDI) lookups.
- Reuse connection and session objects for high throughput applications.
- Use asynchronous consumers over synchronous (polling).
- Specify lazy acknowledgment mode in consumers if duplicate messages are acceptable.
- Choose message types that minimize memory/network use.
- Avoid unnecessary fields in object messages.



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### JMS and Application Design

Asynchronous consumers create less network traffic. Also, asynchronous consumers use fewer threads. An asynchronous consumer does not use a thread while it is inactive. A synchronous consumer consumes a thread for the duration of its receive call. As a result, a thread can remain idle for long periods, especially if the call specifies a blocking timeout.

Because sends are generally faster than receives, consider reducing the overhead associated with receives by deferring acknowledgment of messages until several messages have been received and can be acknowledged collectively. Deferment of acknowledgments is not likely to improve performance for nondurable subscriptions, however, because of internal optimizations already in place.

You may improve the performance of sending large messages traveling across Java Virtual Machine (JVM) boundaries and help conserve disk space by specifying the automatic compression of any messages that exceed a user-specified threshold size. Message compression can help reduce network bottlenecks by automatically reducing the size of messages sent across network wires. Compressing messages can also conserve disk space when storing persistent messages in file stores or databases.

## JMS and Application Design

- Use topics only when the publish/subscribe model is truly a requirement.
- Use simple message selectors for topic consumers, or alternatively use multiple destinations.
- Avoid sending noncritical messages in persistent mode or consuming them with durable subscribers.
- Avoid embedding large amounts of data in standard message headers and user-defined properties.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### JMS and Application Design (continued)

When you are starting to design your application, it is not always immediately obvious whether it would be better to use a topic or queue. In general, you should choose a topic only if the same message must be replicated to multiple consumers and/or a message should be dropped if there are no active consumers that would select it.

Using a selector is expensive. This consideration is important when you are deciding where in the message to store application data that is accessed via JMS selectors.

When designing an application, make sure that you specify that messages will be sent in nonpersistent mode unless a persistent QoS is required. It is recommended that you use the nonpersistent mode, because unless synchronous writes are disabled, a persistent QoS almost certainly causes a significant degradation in performance.

Instead of user-defined message properties, consider using standard JMS message header fields or the message body for message data. Message properties incur an extra cost in serialization and are more expensive to access than standard JMS message header fields. Also, avoid embedding large amounts of data in the properties field or the header fields; only message bodies are paged out when paging is enabled. Consequently, if user-defined message properties are defined in an application, avoid the use of large string properties.

## JMS Quotas Review

- A JMS quota is a reusable policy that:
  - Is added to a JMS module and assigned to destinations
  - Limits the maximum number or size of messages that are stored and awaiting delivery
  - Helps avoid a server running out of memory
- Destinations can use dedicated quotas or share them like a resource pool.
- Destinations that are not assigned a quota share the quota configured on the target JMS server.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### JMS Quotas Review

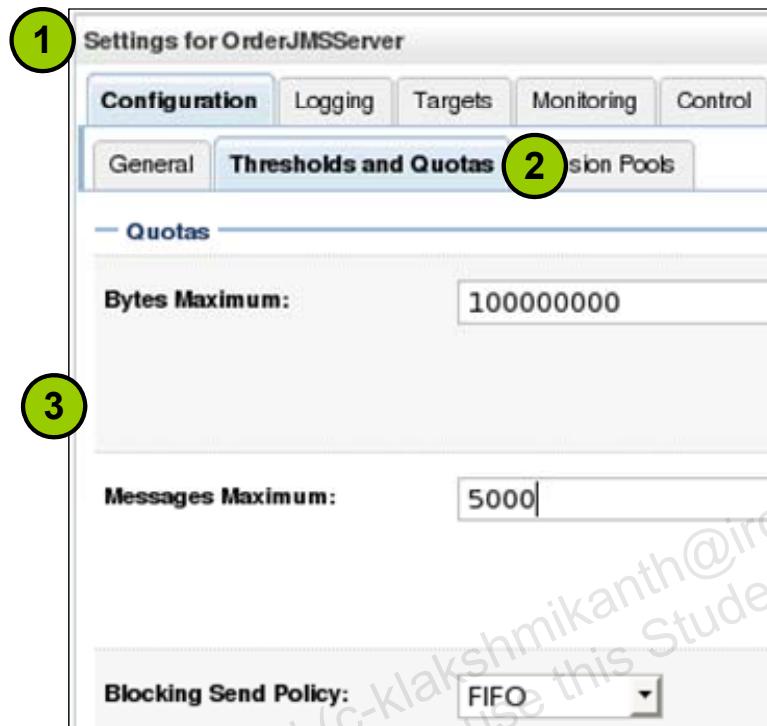
A quota is a named configurable JMS module resource, although quota parameters are also available for entire JMS servers as well. It defines a maximum number of messages and bytes, and is then associated with one or more destinations and is responsible for enforcing the defined maximums. Multiple destinations referring to the same quota share available quota according to the sharing policy for that quota resource.

The Quota parameter of a destination is used to associate it with quota resource to enforce quota for the destination. This value is dynamic, so it can be changed at any time.

In some cases, there will be destinations that do not configure quotas. JMS server quotas allow JMS servers to limit the resources used by these quota-less destinations. All destinations that do not explicitly set a value for the Quota attribute share the quota of the JMS server where they are deployed. However, note that quota resources are not assigned to JMS servers as with destinations. Instead, quota parameters are defined as part of the JMS server itself.

To begin tuning your quota settings, assume that an average message requires 512 bytes of storage, and configure the Bytes Maximum attribute to reflect the maximum memory consumption that your server can tolerate. This value will vary depending on how many other services and applications are competing for memory on the same server.

## Configuring a JMS Server Quota



ORACLE

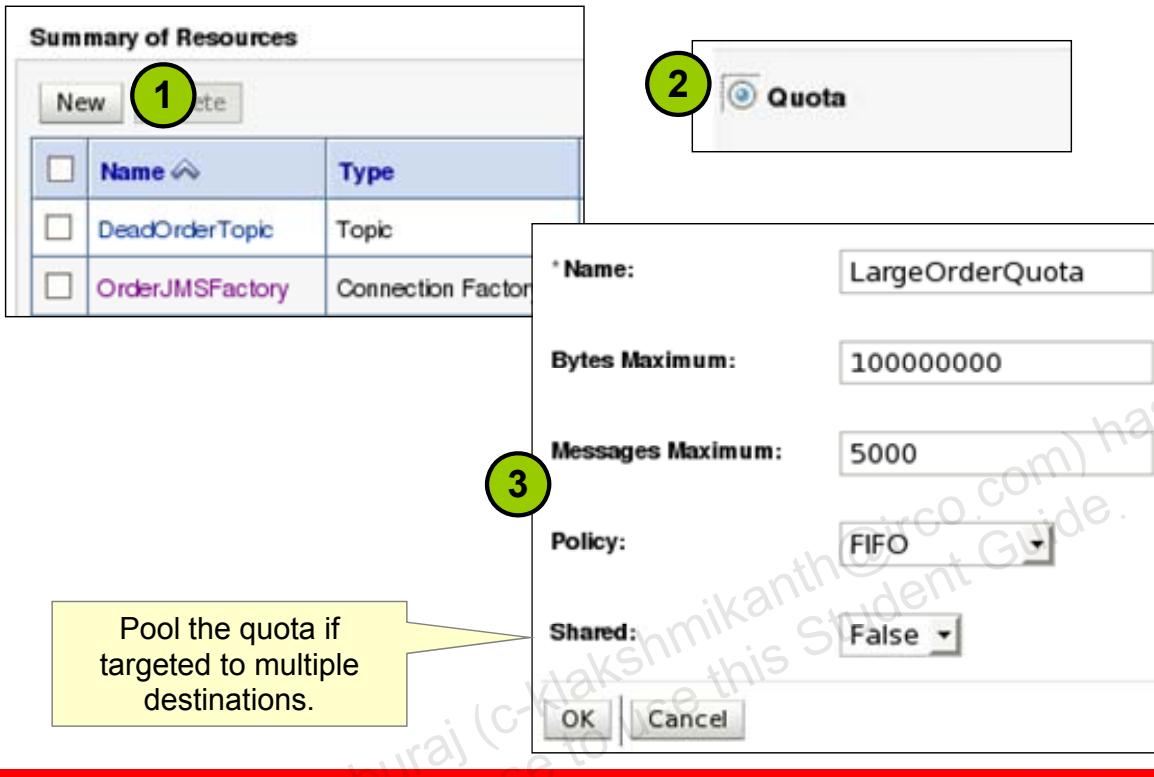
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Configuring a JMS Server Quota

To set quotas on an entire JMS server:

1. Edit an existing JMS server
2. Click "Thresholds and Quotas."
3. Edit the following attributes and click Save:
  - **Bytes Maximum:** The total number of bytes that can be stored by destinations that use this JMS server
  - **Messages Maximum:** The total number of messages that can be stored by destinations that use this JMS server

# Creating a Destination Quota



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Creating a Destination Quota

JMS quotas are used to control the allotment of system resources available to destinations. For example, you can specify the number of bytes or messages that a destination is allowed to store, or specify whether the quota can be shared by all destinations that refer to it. A value of zero means that no messages can be placed on a destination without exceeding the quota. A value of -1 prevents WebLogic Server from imposing a limit.

To create a JMS quota:

1. In the JMS Modules table, click the name of JMS module in which you want to configure the quota resource. Then click the New button in the “Summary of Resources” table.
2. Select Quota from the list of JMS resource types, and click Next.
3. Edit the following attributes and click OK:
  - **Name:** The name of this quota object
  - **Bytes Maximum:** The total number of bytes that can be stored in a destination that uses this quota
  - **Messages Maximum:** The total number of messages that can be stored in a destination that uses this quota
  - **Shared:** Indicates whether this quota is shared by multiple destinations that refer to it. If disabled, the quota object behaves as a template.

## Send Timeout

- If a quota is met, producers receive an exception.
- To avoid situations in which producers continually retry sending messages, tune the **Send Timeout** (ms).
- Connection factories will delay sending the exception, causing producers to wait or “block.”



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Send Timeout

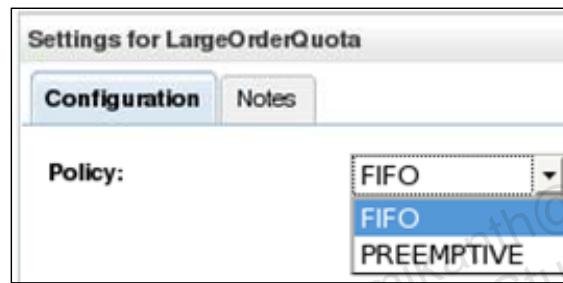
Blocking producers during quota conditions can dramatically improve the performance of applications and benchmarks that continuously retry message sends on quota failures. The Send Timeout connection factory attribute provides more control over message send operations by giving message producers the option of waiting a specified length of time until space becomes available on a destination. For example, if a producer makes a request and there is insufficient space, the producer is blocked until space becomes available, or the operation times out.

In the Send Timeout field, enter the amount of time, in milliseconds, a sender will block messages when there is insufficient space on the message destination. When the specified waiting period ends, one of the following results will occur:

- If sufficient space becomes available before the timeout period ends, the operation continues.
- If sufficient space does not become available before the timeout period ends, you receive a resource allocation exception.

## Quota Blocking Policies

- By default, multiple producers waiting for available quota will be served in the order they attempted to send messages (FIFO).
- Alternatively, smaller messages can be serviced before larger ones as space becomes available.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Quota Blocking Policies

Quota resources and JMS servers provide a blocking send policy attribute to control how message producers compete for space on a destination that has exceeded its message quota. For quota resources, the attribute's name is Policy, while on JMS servers use the Blocking Send Policy attribute found under the Configuration > “Thresholds and Quotas” tab.

Select one of the following options:

- **FIFO:** All send requests for the same destination are queued up one behind the other until space is available. No send request is permitted to complete when another send request is waiting for space before it.
- **Preemptive:** A send operation can preempt other blocking send operations if space is available. That is, if there is sufficient space for the current request, then that space is used even if there are previous requests waiting for space.

# Thresholds and Flow Control Review

- A threshold:
  - Can be configured on a destination or entire JMS server
  - Defines a lower and upper boundary based on the number or size of messages
  - Can trigger log messages and/or flow control
- Flow control:
  - Is enabled on a connection factory
  - Is activated when the upper threshold is exceeded and deactivated for the lower one
  - Introduces delays to slow down producers and keep the server from becoming swamped



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Thresholds and Flow Control Review

High and low thresholds are used to signal periods of load and/or congestion. Administrators can set thresholds for both messages and bytes, and on both JMS servers and destinations. Exceeding these thresholds will trigger events, such as generating log messages and starting message flow control.

With the Flow Control feature, you can direct a JMS server or destination to slow down message producers when it determines that it is becoming overloaded. Specifically, when either a JMS server or its destinations exceeds its specified byte or message threshold, it becomes armed and instructs producers to limit their message flow (messages per second). Producers will limit their production rate based on a set of flow control attributes configured for producers via the JMS connection factory.

As producers are slowed down, the threshold condition gradually corrects itself until the server/destination is unarmed. At this point, a producer is allowed to increase its production rate, but not necessarily to the maximum possible rate. In fact, its message flow continues to be controlled (even though the server/destination is no longer armed) until it reaches its prescribed flow maximum, at which point it is no longer flow controlled.

# Configuring Thresholds

The image contains two screenshots of the Oracle WebLogic Server Administration Console. Both screenshots show the 'Thresholds and Quotas' tab selected under the 'Configuration' section.

**Screenshot 1: Settings for OrderQueue**

Bytes Threshold High:	1000000000
Bytes Threshold Low:	5000000

**Screenshot 2: Settings for OrderJMSServer**

General	Thresholds and Quotas	Session Pools
---------	-----------------------	---------------

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Configuring Thresholds

For all threshold attributes, the default value is `java.lang.Long.MAX_VALUE`, which disables logging and flow control events for the destination or JMS server.

To configure thresholds for a destination:

1. In the JMS Modules table, click the JMS module that contains the configured destination. In the selected JMS module's "Summary of Resources" table, click the queue or topic that you want to configure.
2. Click Configuration > "Thresholds and Quotas." The available attributes are Bytes Threshold High, Bytes Threshold Low, Messages Threshold High, Messages Threshold Low, and Maximum Message Size.

To configure thresholds for a JMS server:

1. In the JMS Servers table, select a JMS server
2. Click Configuration > "Thresholds and Quotas." The available threshold attributes are the same as for destinations.

# Message Compression

- Messages that exceed a given size can automatically be compressed before being sent from producers or delivered to consumers.
- Message compression is:
  - Configured by using connection factories
  - Transparent to producers and consumers
  - Not triggered when clients and server are in the same JVM



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Message Compression

A message compression threshold can be set programmatically by using a JMS API extension or administratively by using the Default Compression Threshold attribute on a connection factory. This attribute specifies the number of bytes for a serialized message body so any message that exceeds this limit will trigger message compression when the message is sent or received by the JMS message producer or consumer.

Messages are compressed by using GZIP. Compression only occurs when message producers and consumers are located on separate server instances where messages must cross a JVM boundary, typically across a network connection when WebLogic domains reside on different machines. Decompression automatically occurs on the client side and only when the message content is accessed. However, using message selectors or filters on consumers to retrieve compressed messages can cause decompression because the message body must be accessed to filter them.

# Scan Expiration Interval

- Expired messages:
  - Are those that have not been consumed and whose time-to-live (TTL) header value has elapsed
  - Are simply deleted by default
  - Can also be written to a log or forwarded to an error destination
- The **Scan Expiration Interval** (seconds) determines how often a JMS server checks for expired messages.



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

ORACLE

## Scan Expiration Interval

Active message expiration ensures that expired messages are cleaned up immediately. Moreover, expired message auditing gives you the option of tracking expired messages, either by logging when a message expires or by redirecting expired messages to a defined error destination.

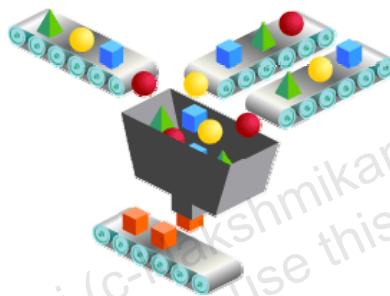
Use the **Expiration Policy** destination attribute to define an alternate action to take when messages expire: discard the message, discard the message and log its removal, or redirect the message to an error destination. Also, if you use JMS templates to configure multiple destinations, you can use the **Expiration Policy** field to quickly configure an expiration policy on all your destinations.

The logging option for message expiration has been deprecated in this release of WebLogic Server. In its place, Oracle recommends using the standard JMS message logging feature.

Messages are not necessarily removed from the system at their expiration time, but they are removed within a user-defined number of seconds. Using the **Scan Expiration Interval** field, enter the amount of time, in seconds, that you want the JMS server to pause between its cycles of scanning its destinations for expired messages to process. To disable active scanning, enter a value of 0 seconds. Expired messages are passively removed from the system as they are discovered.

# Message Ordering

- By default, consumers may not receive messages in the order in which they were produced, due to factors such as:
  - Message priority, time-to-deliver and sort settings
  - Transactional boundaries and rollbacks
  - Consumer acknowledgment mode
- This issue can require complex filtering and collating logic in consumers, which can affect performance.



ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Message Ordering

While the JMS specification provides an ordered message delivery, it does so in a very strict sense. It defines order between a single instance of a producer and a single instance of a consumer, but does not take into account the following common situations:

- Many consumers on one queue
- Multiple producers within a single application acting as a single producer
- Message recoveries or transaction rollbacks where other messages from the same producer can be delivered to another consumer for processing
- Use of filters and destination sort keys

As a result, consumers that need to guarantee message ordering must employ some complex and often expensive design patterns, such as:

- A dedicated consumer with a unique selector per each subordering
- A new destination per subordering, one consumer per destination

## Unit of Order (UOO)

- With WebLogic's Unit of Order feature:
  - Producers provide a unique ID to the server
  - Multiple producers can use a common ID and act as a single producer
  - Messages are consumed sequentially in the order they were produced
- UOO IDs can be set programmatically, configured using a connection factory, or assigned automatically.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

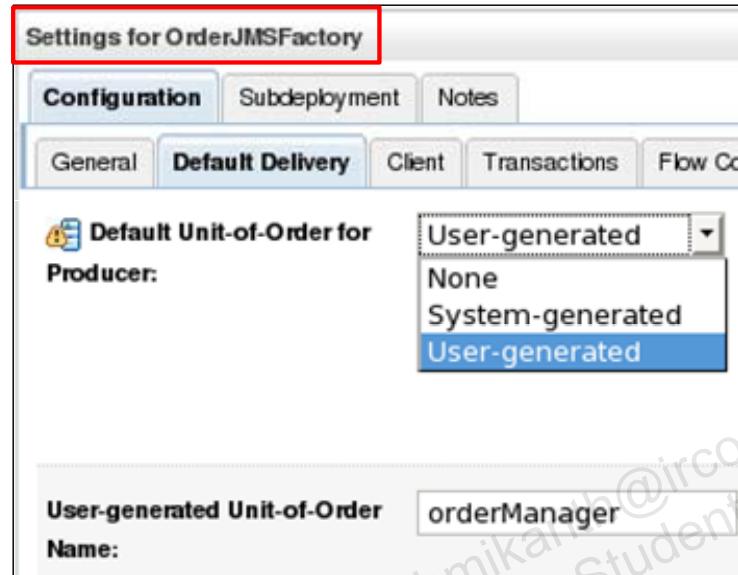
### Unit of Order (UOO)

The WebLogic Server UOO feature enables a message producer or group of message producers acting as one to group messages into a single unit that is processed sequentially in the order in which the messages were created. The message processing of a single message is complete when a message is acknowledged, committed, recovered, or rolled back. Until message processing for a message is complete, the remaining unprocessed messages for that UOO are blocked. UOO is ideal for applications that have strict message ordering requirements. UOO simplifies administration and application design and in most applications improves performance.

Member messages of a UOO are delivered to queue consumers sequentially in the order in which they were created. The message order within a UOO will not be affected by sort criteria, priority, or filters. However, messages that are uncommitted, have a Redelivery Delay, or have an unexpired Time To Deliver will delay messages that arrive after them.

A queue that has several messages from the same UOO must finish processing all of them before they can be delivered to any queue consumer or before the next message can be delivered to the queue.

## Connection Factory UOO



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Connection Factory UOO

Producer UOO names can be set programmatically by the application or can use a name provided by the JMS server. Administrators can explicitly set a default UOO name at the connection factory level or can configure a connection factory or destination to automatically generate a name for producers that have none assigned.

**Default UOO for Producer:** The value System-generated indicates that WebLogic Server automatically generates a UOO name. User-generated indicates that the UOO name comes from the name specified in the subsequent field. If None is selected, no message ordering is enforced.

**User-generated UOO Name:** Specifies the UOO name when the previous attribute's value is set to User-generated

Many applications need an even more restricted notion of a group than that provided by the UOO feature. If this is the case for your applications, WebLogic JMS provides the Unit-of-Work (UOW) capability, which allows applications to send JMS messages, identify some of them as a group, and allow a JMS consumer to process them as such. For example, a JMS producer can designate a set of messages that need to be delivered to a single client without interruption, so that the messages can be processed as a unit. Further, the client will not be blocked waiting for the completion of one unit when there is another unit that is already complete.

# Quiz

Name the two types of blocking policies for JMS quotas.

- a. Preemptive
- b. FIFO
- c. Throttled
- d. LRU
- e. Acknowledge

 ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

**Answer: a, b**

## Quiz

What WLS JMS feature can be used to guarantee the sequence in which messages are consumed?

- a. Imported Destination
- b. Quota Blocking Policy
- c. Unit of Order
- d. Scan Expiration Interval



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

**Answer: c**

## Summary

In this lesson, you should have learned how to:

- List several techniques that can improve JMS performance
- Set quotas and thresholds on JMS servers or destinations
- Tune blocking policies for exceeded quotas
- Tune expired message processing
- Explain WebLogic's Unit of Order feature

Unauthorized reproduction or distribution prohibited. Copyright© 2011, Oracle and/or its affiliates.

LakshmiKanth Kemburaj (c-klakshmikanth@irc0.com) has a  
non-transferable license to use this Student Guide.

# JMS Store and Forward

ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

# Objectives

After completing this lesson, you will be able to:

- Create Store and Forward (SAF) agents, remote contexts, and imported destinations
- Handle expired or failed messages for SAF
- Suspend and resume SAF agents

## WebLogic Store and Forward (SAF)

The store-and-forward feature:

- Provides a mechanism for passing Java Message Service (JMS) messages between two destinations on separate machines
- Is transparent to both JMS consumers and producers
- Uses a persistent store to provide reliability
- Can span multiple WebLogic Server domains
- Can only share messages between WebLogic Server instances, not third-party JMS providers
- Supports Unit of Order to guarantee sequential message delivery



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### WebLogic Store and Forward (SAF)

SAF is an asynchronous communication mechanism. Through SAF, a producer on a local sending endpoint can send a message to a consumer on a remote receiving endpoint, regardless of the availability of the remote endpoint. If the remote endpoint is not available when the producer sends a message, the message is stored in a persistent store or in the memory of the local sending endpoint and forwarded to the remote endpoint when it is available.

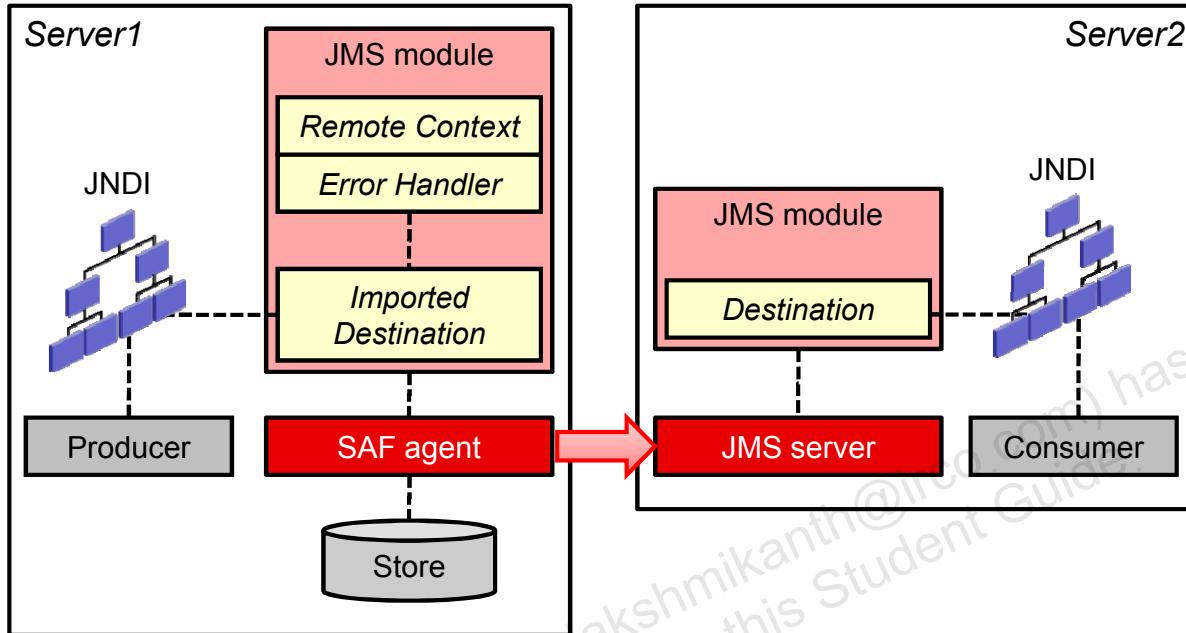
SAF enables message delivery to applications that are distributed across WebLogic Server instances. For example, application A runs on a local WebLogic Server and communicates with application B running on a remote WebLogic Server. Applications A and B are often not connected to each other because of network problems. Therefore, if application A sends a message to application B by using the WebLogic SAF service, the message is stored either in the persistent store of the local WebLogic Server or in the memory of the local WebLogic Server. The message is forwarded when application B is available.

JMS SAF is transparent to JMS applications. Existing JMS applications can take advantage of this feature without any code changes. In fact, you only need to configure imported JMS destinations within JMS modules, which then associate remote JMS destinations to local JNDI names. JMS client code still uses the existing JMS APIs to access the imported destinations.

## **WebLogic Store and Forward (SAF) (continued)**

SAF integrates with the Unit-of-Order feature of WebLogic Server. Unit-of-Order is a mechanism that ensures that messages sent asynchronously using SAF are delivered in the same sequence in which they were created.

# SAF Architecture



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## SAF Architecture

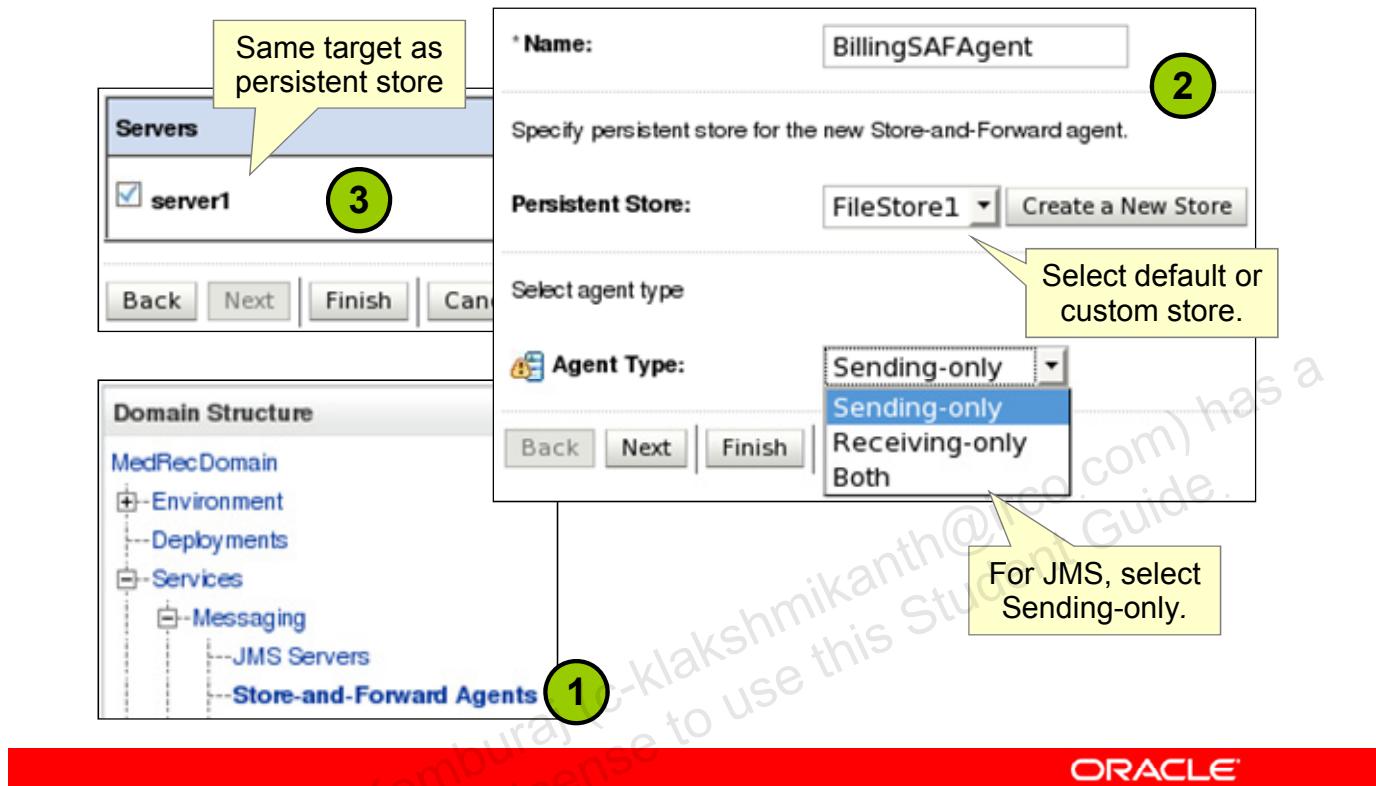
An SAF agent is a configurable object that is similar to a JMS server in that it manages message persistence, paging parameters, and thresholds and quotas. After your JMS SAF resources are configured in a JMS module, the targeted SAF sending agent forwards messages to the receiving side, retransmits messages when acknowledgments do not come back in time, and, if message persistence is required, temporarily stores messages in persistent storage.

An SAF destination (queue or topic) is a local representation of a JMS destination (queue or topic) in a JMS module that is associated with a remote server instance or cluster. Such remote destinations are imported into the local cluster or server instance so that the local server instance or cluster can send messages to the remote server instance or cluster.

A remote SAF context defines the URL of the remote server instance or cluster where the JMS destination is exported from. It also contains the security credentials to be authenticated and authorized in the remote cluster or server. An SAF remote context configuration is required to use imported destinations. A remote SAF context can be reused by multiple SAF-imported destination configurations.

SAF error handling resources define the action to be taken when the SAF service fails to forward messages to a remote destination. SAF error handling resources are not required for imported SAF destinations, but are recommended as a best practice.

# Creating an SAF Agent



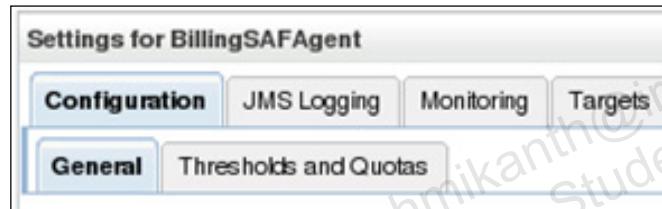
## Creating an SAF Agent

1. In the left panel of the console, expand Services > Messaging, and select Store-and-Forward Agents. Then click New.
2. Edit the following fields and click Next:
  - **Name:** Name for the SAF agent
  - **Persistent Store:** Select a custom persistent store if you want a dedicated store for SAF messages, or click “Create a New Store” to configure a new custom store. If you leave this field set to None, the SAF agent will use the default file store of the target server.
  - **Agent Type:** For JMS SAF, select the Sending-only option. This type of agent stores messages in persistent storage, forwards messages to the receiving side, and retransmits messages when acknowledgments do not come back in time.
3. Select the server instance, cluster, or migratable target where you want to deploy the SAF agent. Then click Finish.

# Configuring an SAF Agent

SAF agents support many of the same attributes as JMS servers, including:

- JMS logging
- Thresholds and quotas
- Message paging



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Configuring an SAF Agent

After you create an SAF agent, you can define basic and advanced general property values for an SAF sending or receiving agent, such as setting message paging defaults, specify delivery retry settings, determine the message window interval, and specify message acknowledgment intervals (WSRM only).

On the Configuration > General tab, the advanced Sending Agent Attributes parameters are near the top, so you must scroll down to the bottom to configure the Receiving Agent Attributes parameters. The latter section applies only to WSRM users, whereas JMS store-and-forward users need only configure fields in the former section.

# Configuring an SAF Agent

<b>Retry Delay Base:</b>	20000	How often to retry failed connections
<b>Retry Delay Maximum:</b>	180000	
<b>Retry Delay Multiplier:</b>	1.0	
<input type="checkbox"/> Pause Incoming Messages At Startup		
<input type="checkbox"/> Pause Forwarding Messages At Startup		The agent must be manually started after server boot.

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

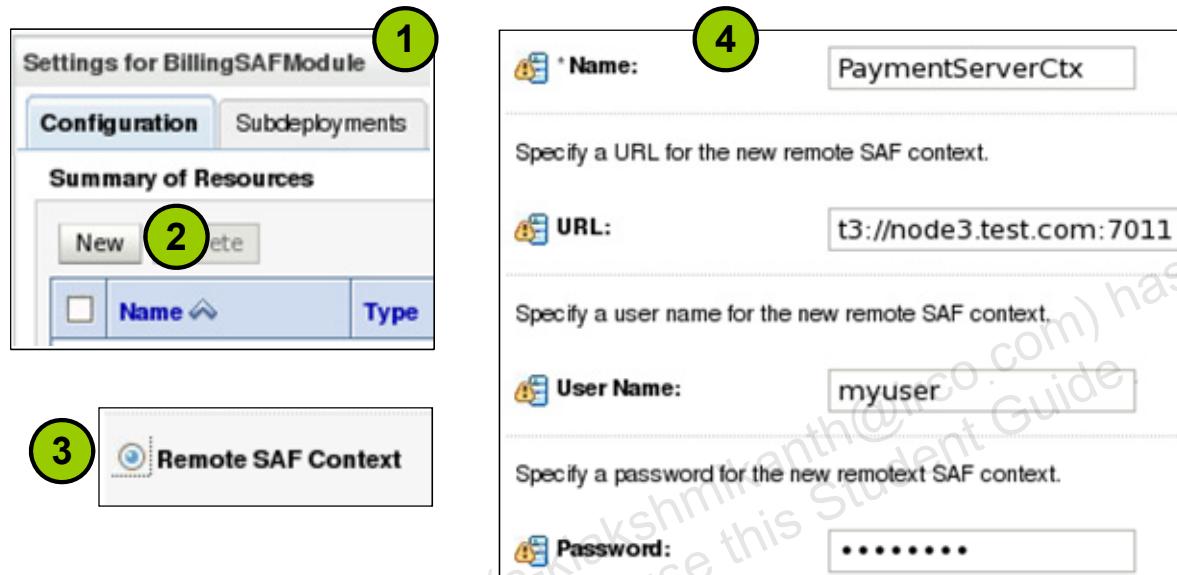
## Configuring an SAF Agent (continued)

The available configuration attributes for an SAF agent include:

- **Retry Delay Base:** The amount of time, in milliseconds, between the original delivery attempt and the first retry
- **Retry Delay Maximum:** The maximum amount of time, in milliseconds, between two successive delivery retry attempts
- **Retry Delay Multiplier:** The factor used to multiply the previous delay time to calculate the next delay time to be used
- **Logging Enabled:** Specifies whether a message is logged in the server log file when a message fails to be forwarded, and when there is no error handling in an SAF application or the application's error handling fails
- **Window Interval:** The maximum amount of time, in milliseconds, that a JMS sending agent will wait before forwarding messages in a single batch. A sending agent will wait to forward the message batch until either: (a) the source destination message count is greater than or equal to the configured Window Size, or (b) it has waited a specified Window Interval, whichever comes first.
- **Pause Incoming Messages At Startup:** When enabled, the sending agent does not accept new messages at server startup. Administrators must then manually resume processing.

# Creating a Remote SAF Context

An SAF context simply defines a connection to remote server.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

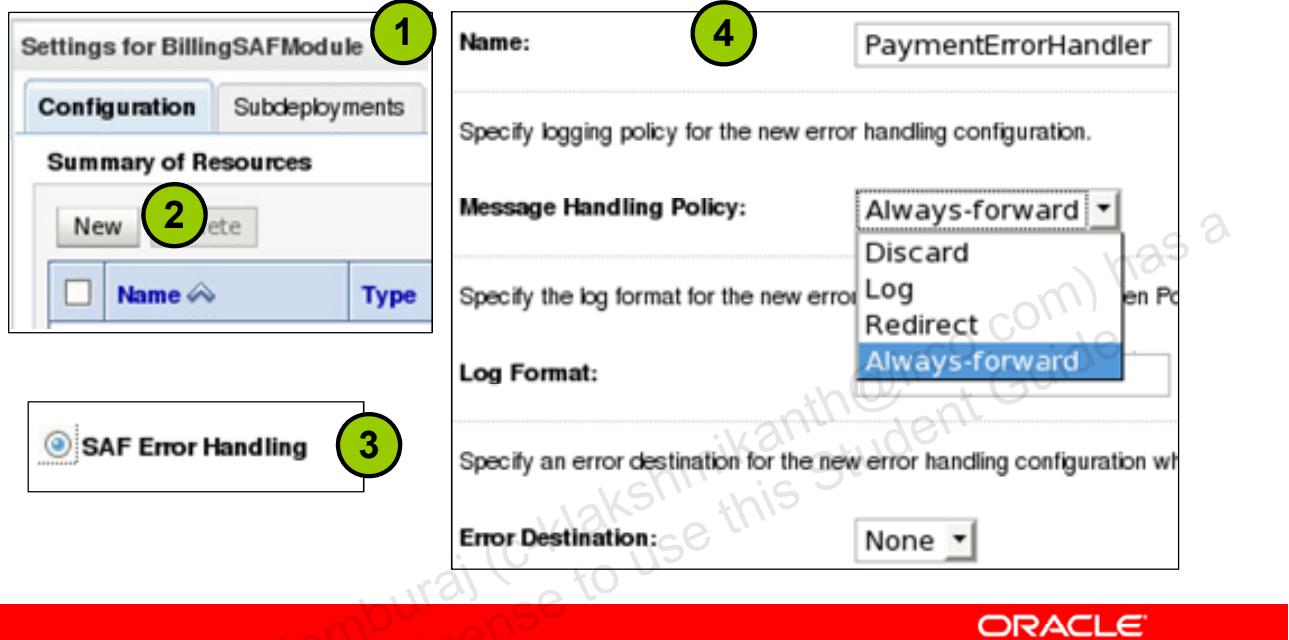
## Creating a Remote SAF Context

A remote SAF context resource specifies the SAF login context that the SAF-imported queue or topic uses to connect to a remote destination. To create an SAF remote context resource:

1. In the JMS Modules table, click the name of JMS module in which to configure the remote SAF context resource.
2. Click New.
3. Select Remote SAF Context and click Next.
4. Edit the properties of the new context and click OK:
  - **Name:** The name for the remote SAF context
  - **URL:** Specifies the URL that the imported queue or topic uses to connect to the remote destination
  - **User Name:** The username that the imported queue or topic gives to the remote destination
  - **Password:** Enter and confirm the password that the imported queue or topic gives to the remote destination

# Creating an SAF Error Handler

Optionally, define an error handler to control how expired and other failed messages are processed.



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

**ORACLE**

## Creating an SAF Error Handler

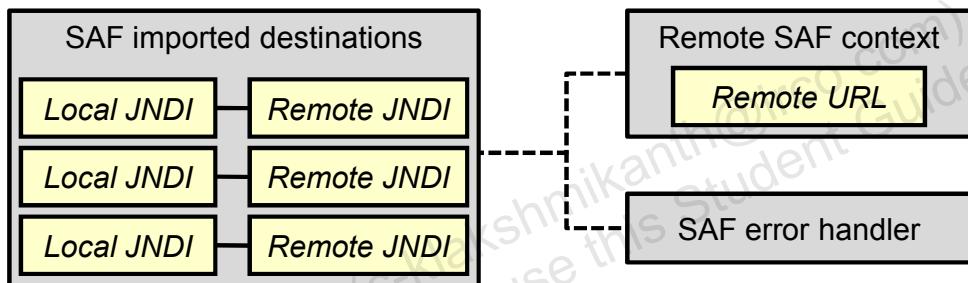
An SAF error handling resource specifies the action to be taken when the SAF service fails to forward messages to a remote destination. To create an SAF error handling resource, perform the following steps:

1. In the JMS Modules table, click the name of JMS module in which to configure the remote SAF context resource.
2. Click New.
3. Select the SAF Error Handling option and click Next.
4. Edit the properties of the new error handling resource and click OK:
  - **Name:** The name for the SAF error handling resource
  - **Message Handling Policy:** Specifies the action to be taken. By default, expired messages are simply removed from the system. The removal is not logged and the message is not redirected to another location. When set to Always-Forward, the SAF agent ignores any Time-to-Live value set on the imported destination and any value set on the message. Therefore, it forwards the message to a remote destination even after it has expired.
  - **Log Format:** Specifies the log format when Message Handling Policy is set to Log
  - **Error Destination:** Specifies an error destination when Message Handling Policy is set to Redirect

# SAF Imported Destinations

An SAF imported destinations resource:

- Is associated with an SAF context and error handler
- Is a container for one or more remote endpoints
- Maps local JNDI names to destinations on the remote server's JNDI service
- Can be targeted to a server or a specific SAF agent



ORACLE

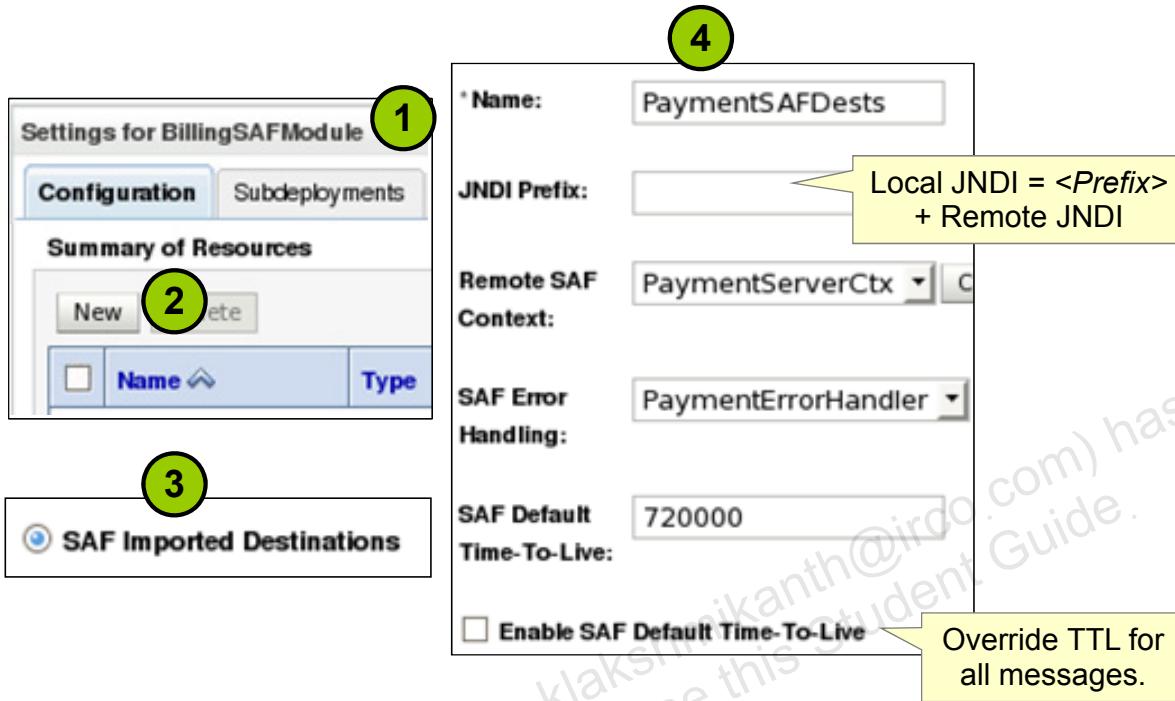
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## SAF Imported Destinations

A collection of imported SAF endpoints (queues or topics) is called an SAF imported destinations resource. Each collection of imported destinations is associated with an SAF remote context. They can also share the same JNDI prefix, default time-to-live (message expiration time), and SAF error handling object.

When a JMS producer sends messages to an SAF destination, these messages are stored on the SAF destination for future delivery. An SAF agent forwards the messages to the remote JMS destination (that the imported destination represents) when the destination is reachable, using the SAF remote context.

# Creating SAF Imported Destinations



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Creating SAF Imported Destinations

1. In the JMS Modules table, click the name of JMS module in which to configure the remote SAF context resource.
2. Click New.
3. Select the SAF Imported Destinations option and click Next.
4. Edit the properties of the new SAF Imported Destinations resource and click Next:
  - **Name:** Enter a name for the SAF imported destinations resource.
  - **JNDI Prefix:** Optionally, specify a prefix that is appended to the remote destination JNDI name to create the local JNDI object. Alternatively, you can use the Local JNDI Name attribute for each remote endpoint (queue or topic) that you add to this resource.
  - **Remote SAF Context:** Select a remote context instance. If necessary, click "Create a New Remote Context" to create one.
  - **SAF Error Handling:** Select an error handling instance. If necessary, click "Create a New Error Handling" to create one.
  - **Enable SAF Default Time-To-Live:** Select to override the expiration time set on JMS messages with the value specified in the SAF Default Time-to-Live field.

## Creating SAF Imported Destinations

The screenshot shows a targeting configuration page. At the top, there's a dropdown menu labeled "Subdeployments" with "DeployToAgent" selected, and a button to "Create a New Subdeployment". Below this, a question asks, "What targets do you want to assign to this subdeployment?". A green circle with the number "5" is overlaid on the page. The "Targets" section contains two lists: "Servers" and "SAF Agents". Under "Servers", there is an unchecked checkbox for "server1". Under "SAF Agents", there is a checked checkbox for "BillingSAFAgent". A yellow callout box points to the "BillingSAFAgent" checkbox with the text: "Target to one or more SAF agents using a subdeployment." At the bottom right, the Oracle logo is visible.

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

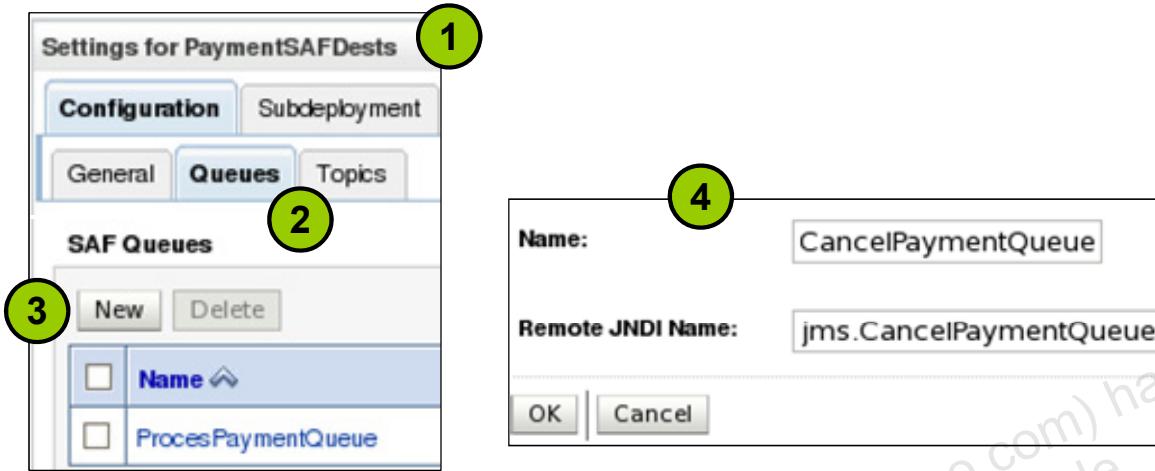
### Creating SAF Imported Destinations (continued)

5. On the targeting page, you can either accept the parent JMS system module's default targets or proceed to an Advanced Targeting page where you can use the subdeployment mechanism for targeting this SAF imported destination.

For basic default targeting, accept the default targets presented in the Targets box and then click Finish. The default targets are based on the parent JMS system module targets.

To select an existing subdeployment for the SAF imported destination, select one from the Subdeployments drop-down list. When a valid subdeployment is selected, its targeted SAF agent(s), server(s), or cluster appear as selected in the Targets box. (A subdeployment with stand-alone destinations can only be targeted to a single JMS server.) Click Finish to add the SAF imported destination to the selected subdeployment. To create a new subdeployment for the SAF imported destination, click the "Create a New Subdeployment" button.

## Adding Remote Endpoints



ORACLE®

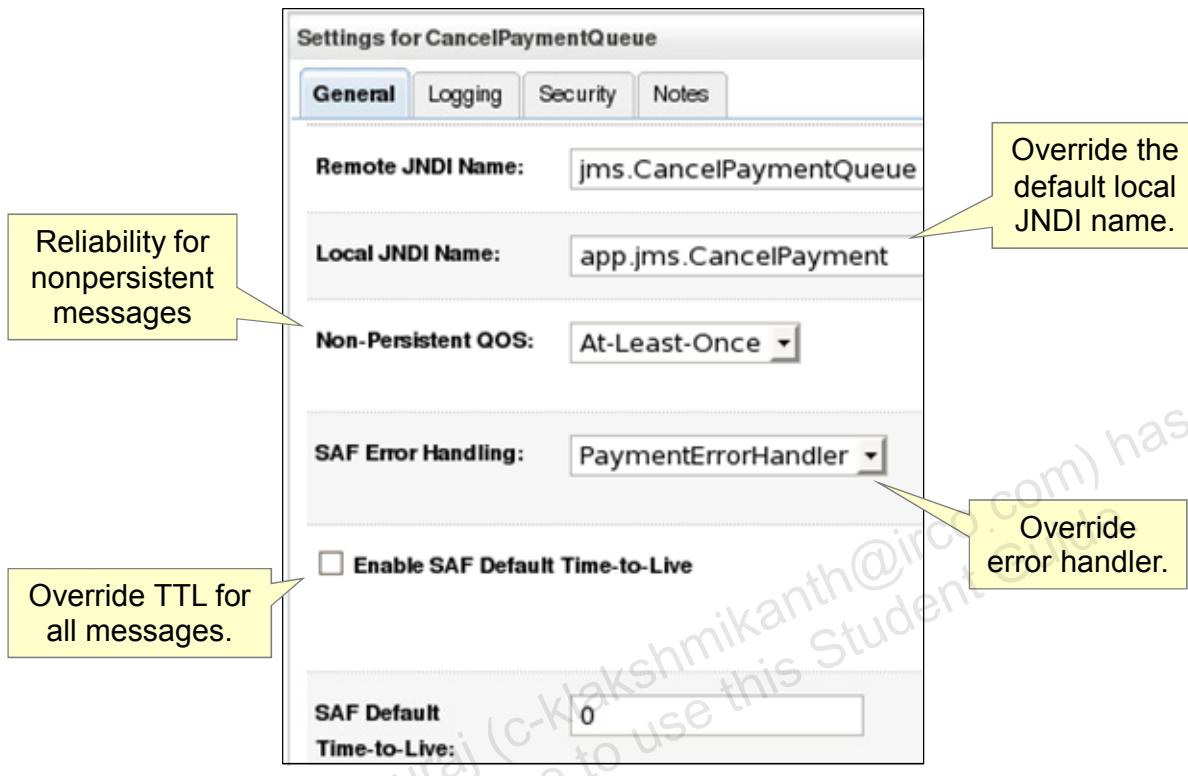
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Adding Remote Endpoints

An SAF queue or topic represents a destination in a remote server or cluster. SAF queues and topics are used for asynchronous and disconnected peer communications. Messages delivered to an SAF queue or topic are temporarily stored for future delivery and are forwarded to a destination on a remote server or cluster when it is reachable.

1. Select an existing SAF Imported Destinations resource within a JMS module.
2. Click Configuration > Queues or Configuration > Topics.
3. Click New.
4. Enter the Name and Remote JNDI Name of the new remote queue or topic. Click OK.

# Configuring Remote Endpoints



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Configuring Remote Endpoints

Edit the new SAF queue or topic to configure additional parameters, including selecting a quality-of-service for nonpersistent messages, choosing an error handling policy, or selecting how Unit-of-Order messages should be routed. The available parameters include:

- **Local JNDI Name:** Specifies the local JNDI name used to access this remote destination.
- **Non-Persistent QOS:** The value Exactly-Once indicates that messages will be forwarded to the remote side once and only once except for any occurrence of server crashes. At-Least-Once indicates that messages will be forwarded to the remote side at least once. No message will be lost except for any occurrence of server crashes. However, messages may appear in the remote endpoint more than once.
- **SAF Error Handling:** Specifies the error handling configuration used by this SAF destination. This overrides any error handling configuration set in the containing imported destinations resource.
- **Enable SAF Default Time-to-Live:** Controls whether the Time-to-Live (expiration time) value set on JMS messages will be overridden by the value specified in the SAF Default Time-to-Live field.
- **Message Unit-of-Order Routing:** Specifies the type of routing used to find an SAF agent when you are using the message Unit-of-Order feature. The value Hash indicates that producers use the hash code of a message Unit-of-Order to find an SAF agent.

## Store and Forward WLST Example

Create an SAF context and imported destination, and target to an SAF agent:

```
edit()
startEdit()
jmsModule = getMBean('/JMSSystemResources/BillingSAFModule/
    JMSResource/BillingSAFModule')
safContext = jmsModule.createSAFRemoteContext('PaymentServerCtx')
safLogin = safContext.getSAFLoginContext()
safLogin.setLoginURL('t3://node3.test.com:7011')
safLogin.setUsername('myuser')
safLogin.setPassword('mypassword')

safDest = jmsModule.createSAFIImportedDestinations('PaymentSAFDest')
safDest.setSAFRemoteContext(safContext)
safDest.setSubDeploymentName('DeployToMySAFAgent')

queue = safDest.createSAFQueue('CancelPaymentQueue')
queue.setRemoteJNDIName('jms.CancelPaymentQueue')
queue.setLocalJNDIName('app.jms.CancelPayment')
save()
activate(block='true')
```



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Store and Forward WLST Example

Refer to the documentation on the following MBeans:

- SAFAgentMBean
- JMSSAFMessageLogFileMBean (an attribute of SAFAgentMBean)
- SAFRemoteContextBean
- SAFLoginContextBean (an attribute of SAFRemoteContextBean)
- SAFErrorHandlingBean
- SAFIImportedDestinationsBean
- SAFQueueBean (a subtype of SAFDestinationBean)
- SAFTopicBean (a subtype of SAFDestinationBean)

# Managing SAF Agents

SAF agents support management activities that are similar to those available for JMS servers:

- Pause an entire agent or specific remote endpoints targeted to the agent.
- Pause incoming and forwarding operations individually.
- View, delete, move, export, or import messages.
- Delete all messages.
- Mark all messages as expired.
- View the message log.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Managing SAF Agents

In WebLogic Server, you can perform the following functions on an SAF agent:

- View runtime information for an active SAF agent and perform message management operations on the agent.
- View runtime information for active remote endpoints (destinations) and perform message management operation on them.
- Control the flow of incoming and/or forwarded messages on a remote endpoint.
- Process all pending messages for a remote destination according to the policy specified by the associated error handling configuration and then remove them from the remote endpoint.
- Destroy all conversations and purge all messages on a remote endpoint.

# Managing an SAF Agent

The screenshot shows the 'Settings for MedRecSAFAgent' window with the 'Monitoring' tab selected. Under the 'Statistics' tab, there is a table titled 'SAF Agents(Filtered - More Columns Exist)'. The table has columns: Incoming, Forwarding, Receiving, Name, Messages Received, Messages Current, and Remote Endpoints Current. A context menu is open over the row for 'MedRecSAFAgent', with 'Pause' and 'Resume' options highlighted.

Incoming	Forwarding	Receiving	Name	Messages Received	Messages Current	Remote Endpoints Current
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	MedRecSAFAgent	0	0	2

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Managing an SAF Agent

1. On the Summary of Store-and-Forward Agents page, select the SAF agent that you want to manage or monitor.
2. Click Monitoring > Statistics.
3. A table displays statistics for the SAF agent. For troubleshooting purposes, runtime message activity can be temporarily paused on SAF agents. Select the check box next to the SAF agent, and then click the Incoming, Forwarding, or Receiving (not applicable to JMS SAF agents) button above the table to select a Pause or Resume operation for that button.

The available statistics for an SAF agent include:

- **Remote Endpoints Current:** Specifies the current number of remote endpoints to which this SAF agent has stored and forwarded messages
- **Remote Endpoints Total:** Specifies the number of remote endpoints to which this SAF agent has stored and forwarded messages since the last time the machine was reset
- **Paused for Incoming:** Indicates whether the sending agent is paused for incoming messages
- **Paused for Forwarding:** Indicates whether the sending agent is paused for forwarding messages
- **Messages Current:** Returns the current number of messages. This number includes the pending messages.

# Managing SAF Agent Endpoints

The screenshot shows the 'Settings for MedRecSAFAgent' window with the 'Monitoring' tab selected. The 'Remote Endpoints' tab is highlighted with a red box. Below it, a table lists two remote endpoints:

Incoming	Forwarding	Actions
<input type="checkbox"/>	<input type="checkbox"/> Name ↗	Pause Resume
<input checked="" type="checkbox"/>	MedRec-jms!Svr2SAFDestinations!CancelPaymentQueue@MedRe	
<input type="checkbox"/>	MedRec-jms!Svr2SAFDestinations!ProcessPaymentQueue@MedR	

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Managing SAF Agent Endpoints

1. On the Summary of Store-and-Forward Agents page, select the SAF agent that you want to manage or monitor.
2. Click Monitoring > Remote Endpoints.
3. Apart from pausing and resuming messaging services, you can purge all messages and destroy all conversations for the selected remote endpoint(s). In addition, you can process all pending messages for a remote destination according to the policy specified by the associated SAF error handling resource. Select the remote endpoint(s) and click Expire All.

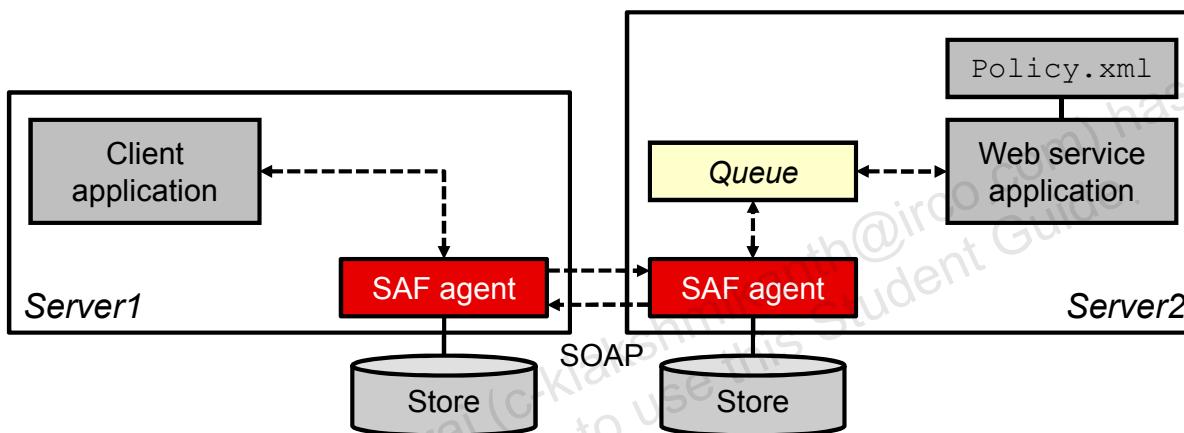
The available statistics for a remote endpoint on an SAF agent include:

- **Paused for Incoming/Forwarding:** The option that indicates whether the sending agent is paused for incoming/forwarding messages currently
- **Downtime High/Total:** The longest/total time, in seconds, that the remote endpoint has not been available since the last time the machine was reset
- **Uptime High/Total:** The longest/total time, in seconds, that the remote endpoint has been available since the last time the machine was reset
- **Last Time Connected:** The last time that the remote endpoint was connected
- Last Time Failed Connect: The last time that the remote endpoint failed to be connected

# Web Services Reliable Messaging (WSRM)

WebLogic Server implements the WSRM specification:

- To guarantee Web service invocations are processed despite communication or server failures
- By employing SAF agents that use Web service protocols (SOAP over HTTP) instead of JMS



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Web Service Reliable Messaging (WSRM)

When you invoke a Web Service synchronously, the invoking client application waits for the response to return before it can continue with its work. In cases where the response returns immediately, this method of invoking the Web Service might be adequate. However, because request processing can be delayed, it is often useful for the client application to continue its work and handle the response later on.

WebLogic Web Services also conform to the WSRM specification, which describes how two Web Services running on different application servers can communicate reliably in the presence of failures in software components, systems, or networks.

A reliable message sequence is used to track the progress of a set of messages that are exchanged reliably between a source and destination. A sequence can be used to send zero or more messages and is identified by a string identifier.

The Web Service client application sends a message for reliable delivery, which is transmitted by the source to the destination. The destination acknowledges that the reliable message has been received and delivers it to the Web Service application. The message may be retransmitted by the source until the acknowledgment is received.

## Web Service Reliable Messaging (WSRM) (continued)

Configuring reliable messaging for a WebLogic Web Service requires standard administrative tasks such as creating JMS servers and SAF agents, as well as Web service developer tasks, such as adding additional WSRM-specific annotations to the Web service source file.

To configure the WebLogic Server instance on which the reliable Web Service is deployed, configure the JMS and SAF resources. You can configure these resources manually or you can use the Configuration Wizard to extend the WebLogic Server domain using a Web Services-specific extension template. Using the Configuration Wizard greatly simplifies the required configuration steps:

1. Create a persistent store (file or JDBC) that will be used by the destination WebLogic Server to store internal Web Service reliable messaging information. You can use an existing one, or the default store that always exists, if you do not want to create a new one.
2. Create a JMS server. If a JMS server already exists, you can use it if you do not want to create a new one.
3. Create a JMS module, and then define a JMS queue in the module. If a JMS module already exists, you can use it if you do not want to create a new one. Target the JMS queue to the JMS server you created in the preceding step. Take note of the JNDI name that you define for the JMS queue, because this must match the name used in the source file that implements your reliable Web service.
4. Create an SAF agent. Set the Agent Type field to Both to enable both sending and receiving agents.

Configuring the WebLogic Server instance on which the Web Service client is deployed involves configuring JMS and SAF resources. Once again, you can configure these manually or with the Configuration Wizard:

1. Create persistent file store.
2. Create an SAF agent. Set the Agent Type field to Both to enable both sending and receiving agents.

# Quiz

Which of the following is NOT a type of resource associated with JMS store and forward?

- a. Remote context
- b. Imported destinations
- c. Error handling
- d. Agent
- e. Adapter



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

**Answer: e**

# Quiz

Which of these is an available attribute for a Remote SAF Context resource?

- a. Local JNDI Name
- b. URL
- c. Agent Type
- d. Time-to-Live

ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

**Answer: b**

## Summary

In this lesson, you should have learned how to:

- Describe the resources that make up WebLogic Server's store and forward (SAF) feature
- Create, manage, and monitor an SAF agent
- Import remote destinations for SAF
- Handle expired or failed messages for SAF
- Describe the relationship between SAF and WSRM



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Practice 11-1: Store and Forward JMS Messages

This practice covers the following topics:

- Creating and monitoring an SAF agent
- Configuring SAF resources in a JMS module
- Defining an SAF context to a remote server
- Forwarding messages to a remote queue
- Verifying message storage across server restarts

Unauthorized reproduction or distribution prohibited. Copyright© 2011, Oracle and/or its affiliates.

LakshmiKanth Kemburaj (c-klakshmikanth@irc0.com) has a  
non-transferable license to use this Student Guide.

# 12

## JMS Message Bridge

ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

# Objectives

After completing this lesson, you will be able to:

- Bridge two different Java Message Service (JMS) providers
- Describe the use of J2EE Connector Architecture (JCA) adapters in JMS bridging
- Configure bridge destinations
- Describe an integration approach for Oracle Advanced Queuing (AQ)

# WebLogic Server Messaging Bridge

A WebLogic messaging bridge:

- Integrates heterogeneous JMS providers
- Forwards messages from one remote destination to another remote destination
- Allows backwards compatibility with older WLS versions
- Communicates with JMS providers via deployable adapters
- Can process messages as an atomic transaction
- Supports connection pooling



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

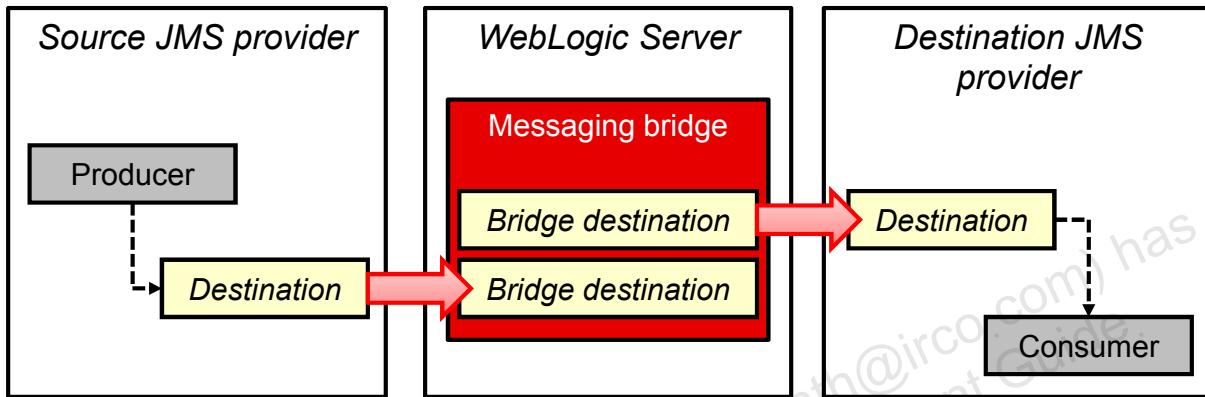
## WebLogic Server Messaging Bridge

The WebLogic messaging bridge is a forwarding mechanism that provides interoperability between WebLogic JMS implementations and between JMS and other messaging products. Use the messaging bridge to integrate your messaging applications between:

- Any two implementations of WebLogic JMS, including those from separate releases of WebLogic Server
- WebLogic JMS implementations that reside in separate WebLogic domains
- WebLogic JMS and a third-party JMS product (for example, MQSeries)

A messaging bridge instance forwards messages between a pair of bridge source and target destinations. These destinations are mapped to a pair of bridge source and target destinations. The messaging bridge reads messages from the source bridge destination and forwards those messages to the target bridge destination. For WebLogic JMS and third-party JMS products, a messaging bridge communicates with source and target destinations using the J2EE Connector Architecture (JCA) resource adapters provided with WebLogic Server.

# Messaging Bridge Architecture



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

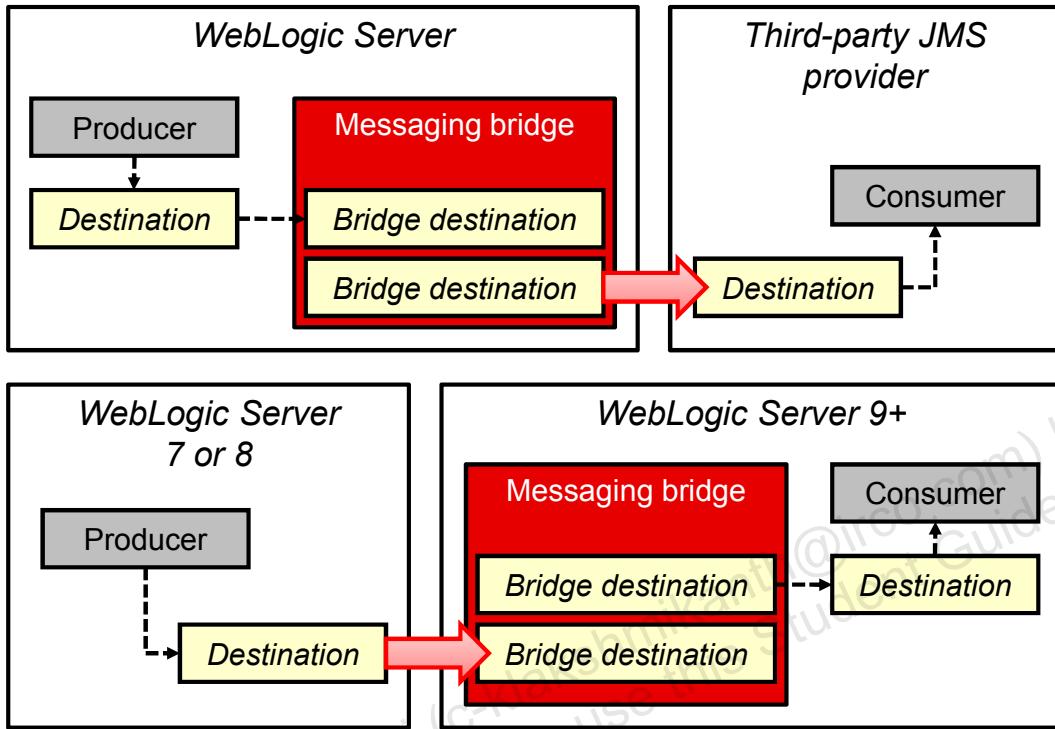
## Messaging Bridge Architecture

A messaging bridge connects two actual destinations that are mapped or attached to bridge destinations. These two destinations are the source destination from which messages are received and the target destination to which the messages are sent.

A messaging bridge instance communicates with the configured source and target bridge destinations. For each mapping of a source destination to a target destination, whether it is another WebLogic JMS implementation or a third-party JMS provider, you must configure a messaging bridge instance.

A messaging bridge uses JCA resource adapters to communicate with the configured source and the target JMS destinations. A messaging bridge is needed to associate both the source and target JMS destinations with a supported resource adapter for the bridge to communicate with the destinations.

# Bridging Scenarios



ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Bridging Scenarios

Messaging bridges provide a way to integrate heterogeneous messaging systems across the enterprise. They are most commonly used to integrate WebLogic JMS with a non-WebLogic JMS provider or to integrate WebLogic JMS with an older version of the product.

When the messaging bridge is used to communicate between two domains running different releases of WebLogic Server, Oracle recommends that the messaging bridge be configured to run on the domain using the latest release of WebLogic Server.

Unique naming rules apply to all WebLogic Server deployments if more than one domain is involved. Therefore, make sure that:

- WebLogic Server instances and domain names are unique
- WebLogic JMS server names are unique names across domains
- If a JMS file store is being used for persistent messages, the JMS file store name must be unique across domains

The messaging bridge cannot provide the “Exactly-once” quality of service (QoS) when the source and target bridge destinations are located on the same resource manager (that is, when the bridge is forwarding a global transaction that is using the XA resource of the resource manager). For example, when using MQSeries, it is not possible to use the same queue manager for the source and target bridge destinations.

# Comparing Bridges to Store and Forward

Feature	SAF	Messaging Bridge
Interoperability with legacy releases of WebLogic Server	No	Yes
Interoperability with third-party JMS products	No	Yes
Performance	Highly optimized for WLS-to-WLS	Generally has higher latency



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Comparing Bridges to Store and Forward

The following sections provide information on when to avoid using the messaging bridge:

- **Sending messages to a local destination:** Send directly to the local destination instead without a bridge.
- **Forward messages between WebLogic 9.0 and higher domain:** Use WebLogic Store-and-Forward (SAF) features instead.
- **Receiving messages from a remote destination:** Instead, use a message-driven EJB or implement a client consumer directly.
- **Environment with low tolerance for message latency:** Messaging bridges increase latency and may lower throughput. Messaging bridges increase latency for messages as they introduce an extra destination in the message path and may lower throughput because they forward messages using a single thread.

# Resource Adapter Review

Resource adapters are:

- Applications defined by the J2EE Connector Architecture (JCA)
- Typically used to facilitate integration with other systems
- Packaged as resource archive files (.rar)
- Deployed to servers just like other application types
- Bound to and accessible from JNDI

Similar to EJBs, adapters use XML deployment descriptors to control transactional behavior, security, and pooling.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Resource Adapter Review

A resource adapter is a system library specific to an Enterprise Information System (EIS) and provides connectivity to an EIS. A resource adapter is analogous to a JDBC driver that provides connectivity to a database management system. The interface between a resource adapter and the EIS is specific to the underlying EIS. It can be a native interface. The resource adapter plugs into an application server, such as WebLogic Server, and provides seamless connectivity between the EIS, application server, and enterprise application.

The structure of a resource adapter and its runtime behavior are defined in deployment descriptors. Programmers create the deployment descriptors during the packaging process, and these become part of the application deployment when the application is compiled. The standard JavaEE descriptor is named `ra.xml`. The WebLogic Server-specific deployment descriptor, `weblogic-ra.xml`, contains elements related to WebLogic Server features such as transaction management, connection management, and security. This file is required for the resource adapter to be deployed to WebLogic Server.

Resource adapters typically include a connection management contract. This enables WebLogic Server to pool connections to an underlying EIS and therefore enables application components to connect to an EIS.

# Bridge Adapters

- WebLogic Server includes two resource adapters for use with bridge destinations.
- Upon creation of a bridge destination, the selected adapter is automatically deployed if not already present.

Bridge Adapter	JNDI Name	Supports XA Transactions?
jms-notran-adp.rar	eis.jms.WLSConnectionFactory JNDINoTX	No
jms-xa-adp.rar	eis.jms.WLSConnectionFactory JNDIXA	Yes



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Bridge Adapters

A messaging bridge uses JCA resource adapters to communicate with the configured source and target JMS destinations. You must associate both the source and target JMS destinations with a supported resource adapter in order for the bridge to communicate with them. The JNDI name for the adapter is configured as part of the resource adapter's deployment descriptor.

Resource adapters for different types of JMS destinations are provided in exploded format or in a .rar file. The exploded format gives you an easy way to modify resource adapter deployment descriptor parameters, such as the max-capacity of the connection factory that specifies the maximum number of connections available for bridge instances.

The supported resource adapters are located in the `WL_HOME\server\lib` directory:

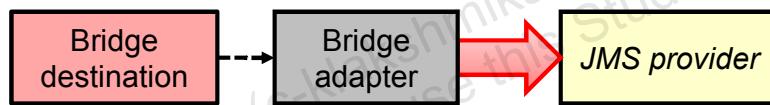
- **jms-xa-adp:** Provides transaction semantics via the XAResource. Used when the required QoS is Exactly-once. This envelops a received message and sends it within a user transaction (XA/JTA).
- **jms-notran-adp:** Provides no transaction semantics. Used when the required QoS is Atmost-once or Duplicate-okay. If the requested QoS is Atmost-once, the resource adapter uses AUTO\_ACKNOWLEDGE mode. If the requested QoS is Duplicate-okay, CLIENT\_ACKNOWLEDGE is used.

## Bridge Adapters (continued)

You may need to modify the capacity of the connection factory associated with each resource adaptor by adjusting the initial-capacity and max-capacity attributes of the weblogic-ra.xml descriptor file. In general, the value of the max-capacity attribute should be at least two times the number of bridge instances. For example, if your environment has up to 10 message bridge instances targeted, a max-capacity attribute setting of 20 in the default configuration is adequate. But, if you increase the number of bridge instances to 15, increase the max-capacity attribute to 30.

# JMS Bridge Destinations

- Create a pair of bridge destinations, each of which defines connectivity to a JMS provider:
  - Bridge adapter
  - Server URL, credentials
  - JNDI initial context factory class
  - Connection factory JNDI name
  - Destination JNDI name
- For non-WLS providers, update your server or adapter CLASSPATH to include the required context factory class and any other supporting libraries.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## JMS Bridge Destinations

A JMS bridge destination instance defines a unique name for a bridge's source and target destinations within a WebLogic domain. Its definition includes the JNDI name of the adapter used to communicate with the specified destination, property information to pass to the adapter such as the URL and connection factory JNDI name, and a username and password.

When bridging to a foreign or third-party JMS provider, you will typically need to make various vendor-specific connection libraries available to the adapter. There are several ways to accomplish this:

- Use the bridge destination attribute named Adapter Classpath.
- Edit your server start scripts and update the CLASSPATH variable.
- Add the required libraries to your domain's /lib folder.

JMS providers may also rely on other non-Java libraries for remote connectivity or protocol/data conversion. These client-side native libraries should be placed in your LD\_LIBRARY\_PATH, SHLIB\_PATH, LIB\_PATH, PATH, or similar appropriate variables.

# Creating a JMS Bridge Destination



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

ORACLE

## Creating a JMS Bridge Destination

You need to configure a JMS bridge destination instance for each actual source and target destination to be mapped to a messaging bridge instance. Therefore, when you finish defining attributes for a source JMS bridge destination, repeat these steps to configure a target JMS bridge destination, or vice versa.

1. In the left pane of the console, expand Services > Messaging > Bridges and select JMS Bridge Destinations.
2. Click New.

# Creating a JMS Bridge Destination

\* Name: **WSMQ\_PaymentProcessing** 3

What is the JNDI name of the adapter you would like to use to communicate with this JMS bridge destination?

**Adapter JNDI Name:** eis.jms.WLSConnectionFactoryJNDIXA  
eis.jms.WLSConnectionFactoryJNDINoTX  
**eis.jms.WLSConnectionFactoryJNDIXA**

What is the CLASSPATH of this JMS bridge destination?

**Adapter Classpath:**

What is the Connection URL of this JMS bridge destination?

**Connection URL:** j.rmi://node13.test.com:2000

What is the JNDI name of the JMS bridge destination connection factory?

\* **Connection Factory JNDI Name:** wsmqXACF

What is the JNDI name of the JMS bridge destination?

\* **Destination JNDI Name:** PaymentQueue

**ORACLE**

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

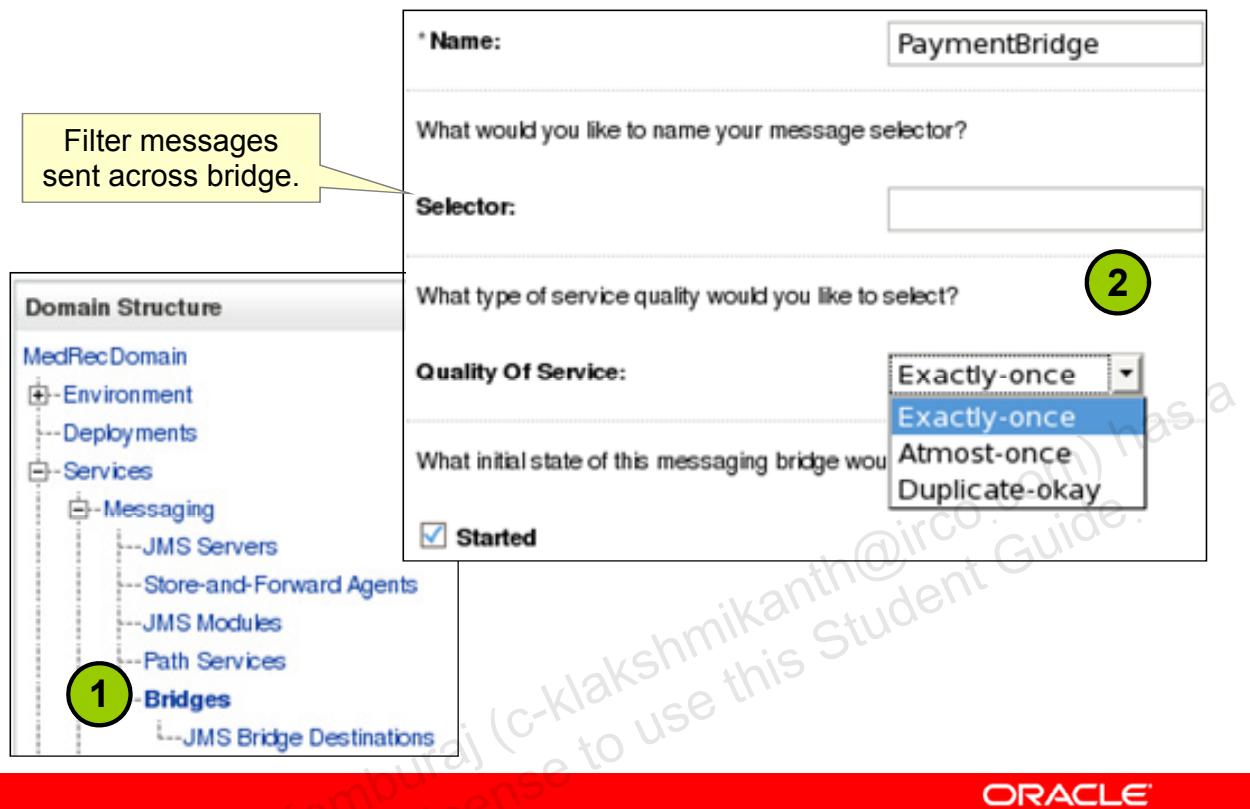
## Creating a JMS Bridge Destination (continued)

3. Define the configuration attributes for a JMS bridge destination and click OK.

When creating or editing a bridge destination, the available attributes include:

- **Name:** The bridge destination name that must be unique across a WebLogic Server domain
- **Adapter JNDI Name:** The JNDI name of the resource adapter that the message bridge will use. WebLogic Server supports two default resource adapters. Based on your selection, the appropriate adapter will be deployed to your domain automatically.
- **Adapter Classpath:** Any classpath additions required to communicate with a third-party JMS provider. Alternatively, modify the server classpath.
- **Connection URL:** The location of the remote provider this bridge destination will connect to
- **Connection Factory JNDI Name:** The JNDI name of the connection factory that this bridge destination will use
- **Destination JNDI Name:** The JNDI name of the JMS destination to which the bridge destination will map
- **Initial Context Factory:** The name of the initial context factory Java class for the JMS bridge destination to use to connect to the remote JNDI service

# Creating a Message Bridge



## Creating a Message Bridge

A messaging bridge instance communicates with the configured source and target bridge destinations.

1. In the left pane of the console, expand Services > Messaging and select Bridges. Then click New.
2. Define the properties that will identify your new messaging bridge instance and click Next:
  - **Name:** The name of the messaging bridge destination
  - **Selector:** A selector for filtering the messages forwarded using the messaging bridge instance
  - **Quality Of Service:** A QoS level for the delivery of the messaging bridge message. You can choose from three levels of QoS:
    - **Exactly-once:** The highest QoS guarantees that a message is sent to the remote endpoint only once
    - **Duplicate-okay:** Guarantees that a message is sent to the remote endpoint, but with the possibility of duplicates

## Creating a Message Bridge (continued)

- **Atmost-once:** The lowest QoS guarantees that each message is sent to the remote endpoint only once, if at all
- **Started:** The option to start the messaging bridge after it is created. The only method available to dynamically start and stop a bridge is this attribute.

# Creating a Message Bridge

3 Select an Existing Source Destination: WLS\_PaymentProcessing ▾ New Destination

Messaging Provider: WebLogic Server 7.0 or higher

WebLogic Server 7.0 or higher

WebLogic Server 6.1

WebLogic Server 5.1

Other JMS

Back Next Cancel

4

5 Select an Existing Target Destination: WSMQ\_PaymentProcessing ▾ New Destination

Messaging Provider: Other JMS

WebLogic Server 7.0 or higher

WebLogic Server 6.1

WebLogic Server 5.1

Other JMS

Back Next Cancel

6

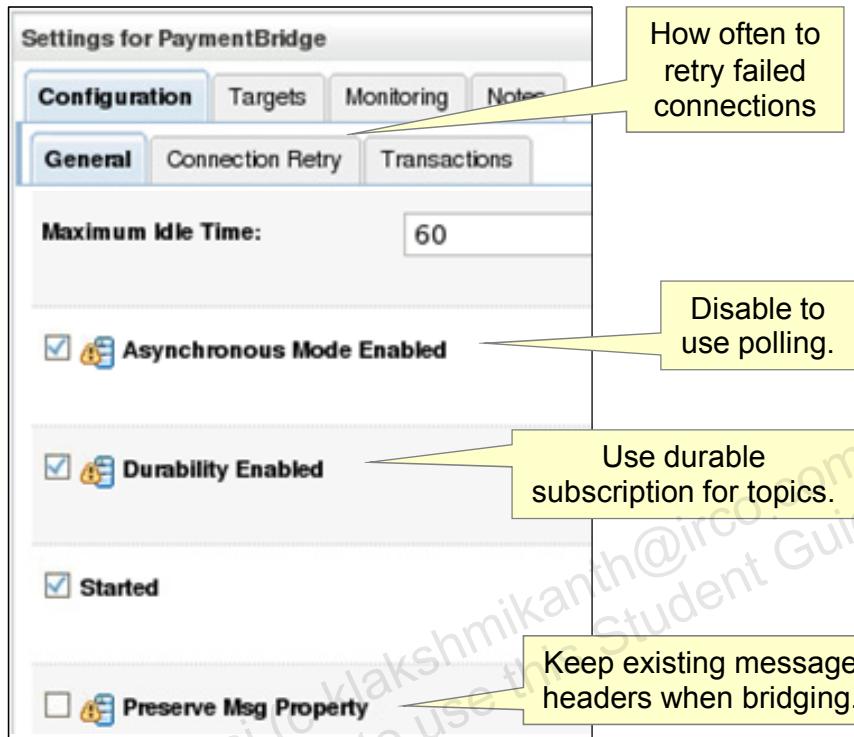
ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Creating a Message Bridge (continued)

3. Select a source destination from the drop-down list. If the drop-down list is empty or does not contain the source destination you need, click New Destination. Click Next.
4. Select the type of JMS provider associated with the source bridge destination. For non-WLS providers, select the option Other JMS. Click Next.
5. Select a target destination from the drop-down list. Click Next.
6. Select the type of JMS provider associated with the target bridge destination. Click Next.
7. Select a target for your messaging bridge instance. Click Next.
8. Review any additional interoperability guidelines and click Finish.

# Configuring a Message Bridge



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Configuring a Message Bridge

The Connection Retry parameters configure the time required to reconnect the messaging bridge with the target. If you set low values for these parameters, it can result in an increase in load over WebLogic Server, whereas if you set high values, it can result in a substantial delay in the delivery of the messages.

**QOS Degradation Allowed:** When enabled, the messaging bridge instance degrades the QoS when the configured QoS is not available. If the QoS is degraded, a log message is delivered to the WebLogic startup window or log file. When not enabled, if the messaging bridge instance cannot satisfy the quality of service requested, an error results and the messaging bridge instance does not start.

**Maximum Idle Time:** In asynchronous mode, this is the longest amount of time a messaging bridge instance stays idle before it checks its connection to the source. In synchronous mode, this is the amount of time the messaging bridge can block on a receive call if no transaction is involved.

## Configuring a Message Bridge (continued)

**Asynchronous Mode Enabled:** Messaging bridge instances that forward in asynchronous mode are driven by the source destination. The bridge instance listens for messages and forwards them as they arrive. When not selected, a bridge instance is forced to work in synchronous mode, even if the source supports asynchronous receiving. For a messaging bridge instance with a QoS of “Exactly-once” to work in asynchronous mode, the source destination has to support the MDBTransaction interface. Otherwise, the bridge automatically switches to synchronous mode if it detects that this interface is not supported by the source destination.

**Durability Enabled:** When enabled and the source topic supports durable subscriptions, the source JMS implementation saves messages that are sent when a messaging bridge instance is not running. When the bridge instance is restarted, these messages are forwarded to the target destination. When not enabled, messages that are sent to the source topic while the bridge instance is down will not be received.

**Preserve Msg Property:** Set to preserve message properties in a message header when a message is forwarded by a bridge instance. When enabled, an incoming nonpersistent message is forwarded by the bridge to the target destination as a nonpersistent message and an incoming persistent message is forwarded to the target destination as a persistent message. However, the Messaging Bridge never discloses a message’s JMSXUserID property across the Messaging Bridge’s boundaries. JMSXUserID is a WebLogic Server-specific, system-generated property that identifies the user sending the message.

## Message Bridge WLST Example

Create and target a message bridge:

```
...
dest2 = create('WSMQ_PaymentProcessing', 'JMSBridgeDestination')
dest2.setInitialContextFactory('org.objectweb.rmi.libs.services.
    registry.jndi.JRMIInitialContextFactory')
dest2.setConnectionURL('j.rmi://node13.test.com:2000')
dest2.setAdapterJNDIName('eis.jms.WLSConnectionFactoryJNDIXA')
dest2.setConnectionFactoryJNDIName('wsmqXACF')
dest2.setDestinationJNDIName('paymentQueue')
dest2.setDestinationType('Queue')
dest2.setUserName('myuser')
dest2.setUserPassword('mypassword')

bridge = create('PaymentBridge', 'MessagingBridge')
bridge.setSourceDestination(dest1))
bridge.setTargetDestination(dest2))
bridge.setQualityOfService('Exactly-once')
bridge.setStarted(true)
bridge.addTarget(getMBean('/Servers/server1'))
...
...
```



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Message Bridge WLST Example

Refer to the documentation on the following MBeans:

- JMSBridgeDestinationMBean (a subtype of BridgeDestinationCommonMBean)
- MessagingBridgeMBean

# Oracle Advanced Queuing (AQ) Overview

Oracle AQ:

- Extends an Oracle database to support a messaging infrastructure
- Is used internally by other Oracle products to asynchronously distribute and receive events
- Includes a DDL to define queues and topics
- Provides a JMS-compliant client interface that wraps a JDBC connection



AQ JMS producers and consumers:

- Utilize an AQ-specific initial context factory
- Specify additional properties to connect to the database

**ORACLE**

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

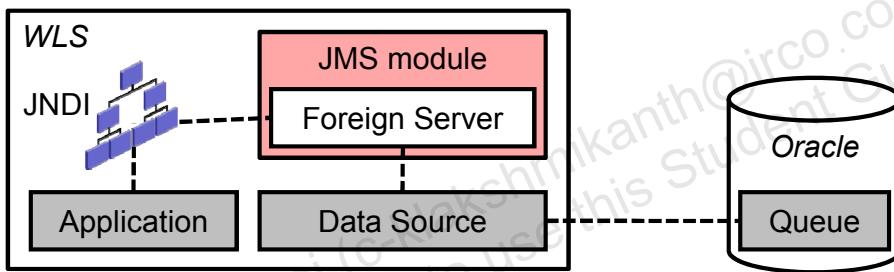
## Oracle Advanced Queuing (AQ) Overview

Oracle Streams AQ provides database-integrated message queuing functionality. It is built on top of Oracle Streams and leverages the functions of Oracle Database so that messages can be stored persistently, propagated between queues on different computers and databases, and transmitted using Oracle Net Services and HTTP(S). Because Oracle Streams AQ is implemented in database tables, all operational benefits of high availability, scalability, and reliability are also applicable to queue data. Standard database features such as recovery, restart, and security are supported by Oracle Streams AQ. You can use database development and management tools such as Oracle Enterprise Manager to monitor queues. Like other database tables, queue tables can be imported and exported. Messages can be queried using standard SQL. This means that you can use SQL to access the message properties, the message history, and the payload. With SQL access you can also do auditing and tracking. All available SQL technology, such as indexes, can be used to optimize access to messages.

Oracle Java Message Service (OJMS) provides a Java API for Oracle Streams AQ based on the JMS standard. OJMS supports the standard JMS interfaces and has extensions to support administrative operations and other features that are not a part of the standard. Oracle JMS interfaces are in the oracle.jms package. To use JMS with clients running outside the database, you must include the appropriate JDBC driver, JNDI jar files, and Oracle Streams AQ jar files in your CLASSPATH.

# AQ Integration Overview

- The SAF and Messaging Bridge features do not support connectivity to Oracle AQ.
- In order for WebLogic applications to access AQ resources as if they were local JMS destinations:
  - Define a JMS Foreign Server and associate it with an existing data source
  - Map local JNDI names to remote AQ resources, similar to SAF



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

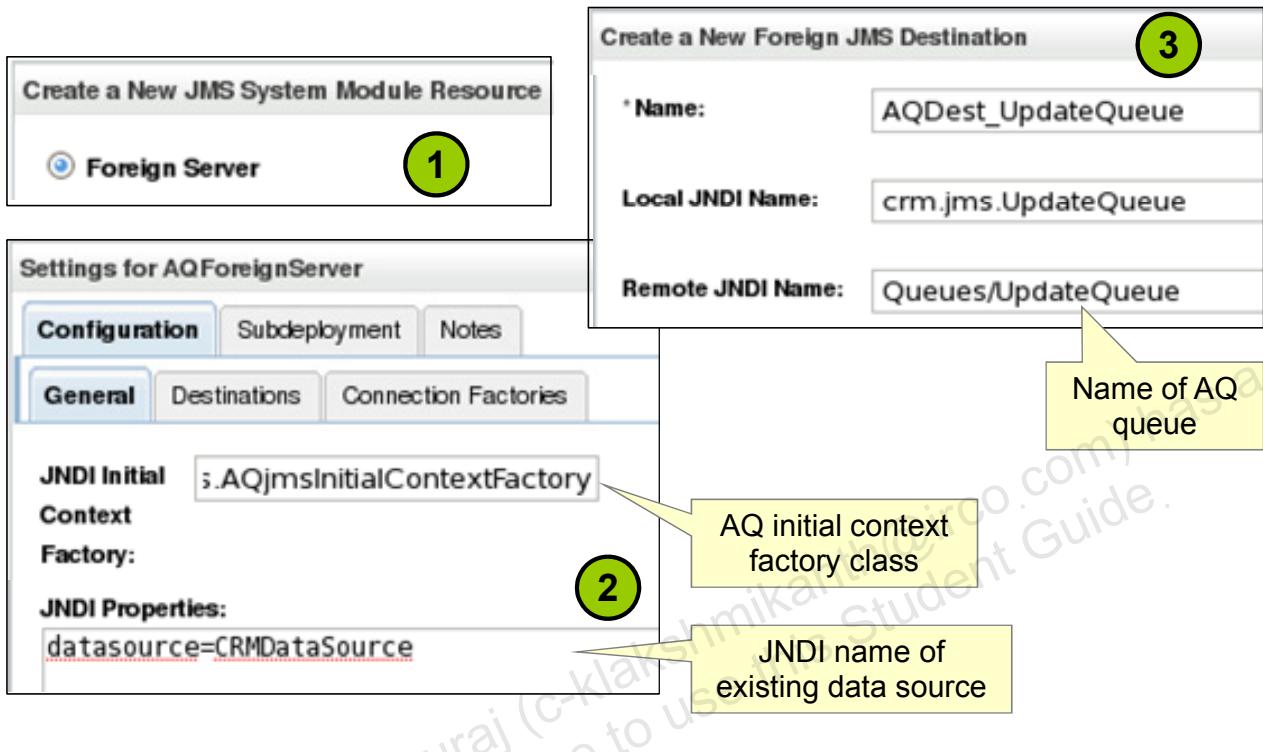
## AQ Integration Overview

Your WebLogic Server installation includes all the necessary classes to communicate with Oracle Streams AQ. No additional files are required in the WebLogic Server classpath. Administrators use the WebLogic JMS Foreign Server framework to allow WebLogic Server applications and stand-alone clients to lookup AQ JMS connection factories and destinations via the standard WebLogic JNDI context and standard JMS APIs. The required references to the AQ data source are configured as part of this framework. Only the Oracle JDBC 11g thin driver, included in your WebLogic Server installation, is supported for this release. Global XA (JTA) transactions and local JMS-transacted session transactions are also supported. As always, global transactions require use of XA-based JMS connection factories.

Local JNDI names are defined for AQ JMS connection factories and destinations as part of the WebLogic JMS Foreign Server configuration. These JNDI names are configured to map to existing AQ connection factories and destinations. In turn, WebLogic Server applications, such as message-driven EJBs, reference the local JNDI names.

AQ does not require the explicit creation of connection factories. Refer to the WLS or AQ documentation for the JNDI names of built-in AQ connection factories. For destinations, AQ requires a specific syntax for JNDI names. If the destination is a queue, the remote JNDI name must be "Queues/<queue name>." If the destination is a topic, the remote JNDI name must be "Topics/<topic name>."

# Creating a Foreign Server for AQ



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Creating a Foreign Server for AQ

1. Select an existing JMS module or create a new one. Click the New button in the Summary of Resources table and then select the option Foreign Server. Click Next. Name the resource and click Next again. Accept the default targets of the parent module and click Finish.
2. Edit the new Foreign Server. For JNDI Initial Context Factory, enter oracle.jms.AQjmsInitialContextFactory. Within JNDI Properties, enter the line datasource=<ds>, where <ds> is the JNDI name of an existing data source associated with your AQ database. Click Save.
3. Click Configuration > Destinations. Click New. Enter the following and click OK:
  - **Name:** The configuration name
  - **Local JNDI Name:** The name that the remote destination will be bound to in the local server's JNDI tree for local applications to access
  - **Remote JNDI Name:** The JNDI name of the remote destination on AQ
4. (Not shown) Click Configuration > Connection Factories. Repeat the same steps that you used to define a remote destination. Enter the JNDI name of the connection factory on AQ and bind it to a local JNDI name.

## Quiz

How many JMS bridge adapters ship with WebLogic Server?

- a. 0
- b. 1
- c. 2
- d. 3
- e. 4

 ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

**Answer: c**

# Quiz

Which of these is an available Message Bridge attribute?

- a. XA Enabled
- b. Connection URL
- c. Local JNDI Name
- d. Quality of Service



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

**Answer: d**

## Summary

In this lesson, you should have learned how to:

- Describe a WebLogic Server messaging bridge
- Contrast WebLogic Server's JMS bridge and SAF features
- Configure bridge destinations
- Assemble a bridge using source and target destinations
- Describe a technique for integrating WebLogic Server JMS with Oracle AQ

## Practice 12-1: Bridge JMS Providers

This practice covers the following topics:

- Defining a bridge destination for WebLogic Server JMS
- Defining a bridge destination for a third-party JMS provider (Glassfish OpenMQ)
- Configuring a message bridge between destinations

Unauthorized reproduction or distribution prohibited. Copyright© 2011, Oracle and/or its affiliates.

LakshmiKanth Kemburaj (c-klakshmikanth@irc0.com) has a  
non-transferable license to use this Student Guide.

# 13

## Server Migration

ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

# Objectives

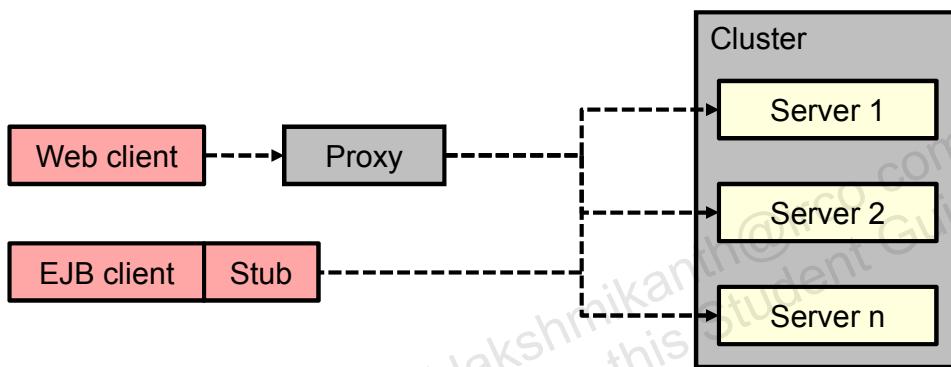
After completing this lesson, you will be able to:

- Configure automatic server migration from one machine to another
- Configure the cluster leasing service
- Configure the node manager to support migration
- Manually migrate a whole server
- Describe the capabilities of WebLogic Operations Control

# WebLogic Clustering Review

A cluster supports additional features that are transparent to applications and clients:

- To provide high availability for applications and services
- To perform load balancing and failover



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## WebLogic Clustering Review

Clustering is configuring a group of Oracle WebLogic Servers to work together to provide client access to the services offered by the servers in the cluster. The cluster appears to a client as one instance, whether the client is a Web client or a Java application. By replicating the services provided by one instance, an enterprise system achieves a fail-safe and scalable environment. Scalability is achieved by balancing the load of incoming requests across the servers in the cluster.

High availability is achieved through the replication of services, so that when one service fails, another service can resume where the first service left off. A cluster uses the redundancy of multiple servers to insulate clients from failures.

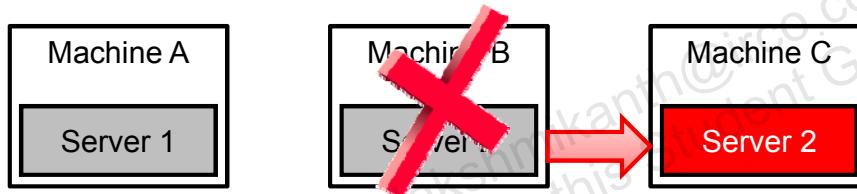
Oracle WebLogic Server provides clustering support for Web applications by replicating the HTTP session state of clients. You can balance the Web application load across a cluster by using an Oracle WebLogic Server proxy plug-in or the external load-balancing hardware.

Failover for Enterprise JavaBeans (EJBs) objects is accomplished using the object's replica-aware stub. When a client makes a call through a replica-aware stub to a service that fails, the stub detects the failure and retries the call on another replica.

# Whole-Server Migration

WebLogic's whole-server migration infrastructure:

- Allows a server on a failed machine to be restarted on another machine
- Requires a running Node Manager on each machine participating in migration
- Requires servers to be members of a cluster
- Supports both manual and automatic failover



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Whole-Server Migration

When a migratable server becomes unavailable for any reason (for example, if it hangs, loses network connectivity, or its host machine fails), migration is automatic. Upon failure, a migratable server is automatically restarted on the same machine if possible. If the migratable server cannot be restarted on the machine where it failed, it is migrated to another machine. In addition, an administrator can manually initiate migration of a server instance.

Node Manager is used by the Administration Server or a stand-alone Node Manager client to start and stop migratable servers and is invoked by the cluster master to shut down and restart migratable servers, as necessary.

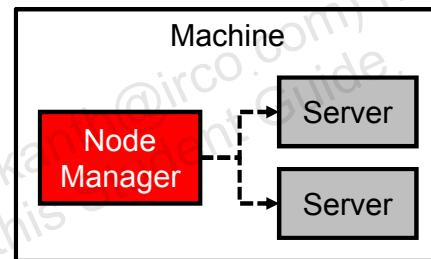
Server migration has the following additional requirements:

- There is no built-in mechanism for transferring files that a server depends on between machines. Using a disk that is accessible from all machines is the preferred way to ensure file availability. If you cannot share disks between servers, you must ensure that the contents of `domain_dir/bin` are copied to each machine.
- You cannot create network channels that have different listen addresses on a migratable server.
- Although migration works when servers are not time-synchronized, time-synchronized servers are recommended in a clustered environment.

# Node Manager Review

## The WebLogic Node Manager:

- Is a stand-alone program (Java or UNIX script options) that runs on each machine in your domain
- Allows administrators to start and kill servers from remote locations
- Periodically checks the availability of servers on the same machine
- Tries to restart failed servers a specified number of times
- Can also restart servers after a failed machine reboots



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Node Manager Review

Server instances in a WebLogic Server production environment are often distributed across multiple domains, machines, and geographic locations. Node manager is a WebLogic Server utility that enables you to start, shut down, and restart Administration Server and managed server instances from a remote location. Although Node Manager is optional, it is recommended if your WebLogic Server environment hosts applications with high availability requirements.

A Node Manager process is not associated with a specific WebLogic domain but with a machine. You can use the same Node Manager process to control server instances in any WebLogic Server domain, as long as the server instances reside on the same machine as the Node Manager process.

From the WebLogic Server Scripting Tool (WLST) or administration console, you can issue commands to Node Manager to start, shut down, suspend, and restart managed server instances and clusters. If a server instance that was started using Node Manager fails, Node Manager automatically restarts it. Finally, you can configure the Node Manager to periodically monitor the health state JMX attribute of its servers and automatically kill and restart servers in failed states.

## **Node Manager Review (continued)**

WebLogic Server provides two versions of Node Manager, Java-based and UNIX script-based, with similar functionality. However, each version has different configuration and security considerations.

## Leasing Service

- To support automatic migration, clusters use a central leasing service.
- Clustered servers periodically report the machine that they are currently using or “leasing.”
- One server in the cluster always assumes the role of master, which:
  - Periodically checks the leasing service for expired leases, indicating server or machine failure
  - Can automatically initiate server migration when the machine or its Node Manager is unavailable
- If cluster heartbeat messages indicate that the current master has failed, a new master is selected.

ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Leasing Service

Leasing is the process WebLogic Server uses to manage services that are required to run on only one member of a cluster at a time. Leasing ensures exclusive ownership of a cluster-wide entity. Within a cluster, there is a single owner of a lease. Additionally, leases can fail over in case of server or cluster failure. This helps to avoid having a single point of failure.

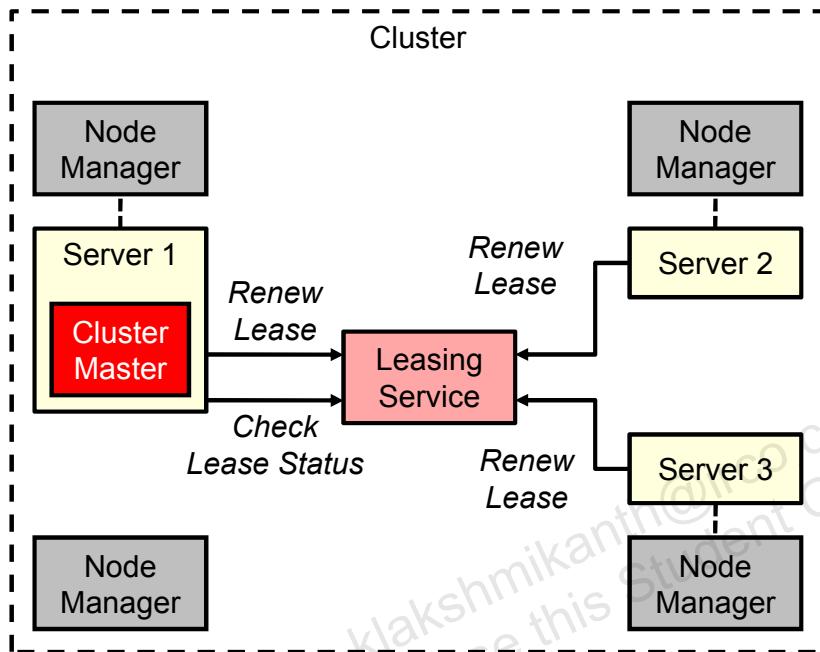
In a cluster that contains migratable servers, one server instance acts as the cluster master. Its role is to orchestrate the server migration process. Any server instance in the cluster can serve as the cluster master. When you start a cluster that contains migratable servers, the first server to join the cluster becomes the cluster master and starts up the cluster manager service.

Cluster masters perform the following activities:

- Issue periodic heartbeats to the other servers in the cluster
- Periodically check the leasing service to verify that each migratable server has a current lease. An expired lease indicates to the cluster master that the migratable server should be restarted.

If unable to restart a migratable server whose lease has expired on its current machine due to the fact that the machine’s Node Manager is unavailable, the cluster master selects a new target machine to run the server on. The cluster master invokes the Node Manager process on the target machine to create a process for the server instance.

## Automatic Server Migration Architecture: No Failure



ORACLE®

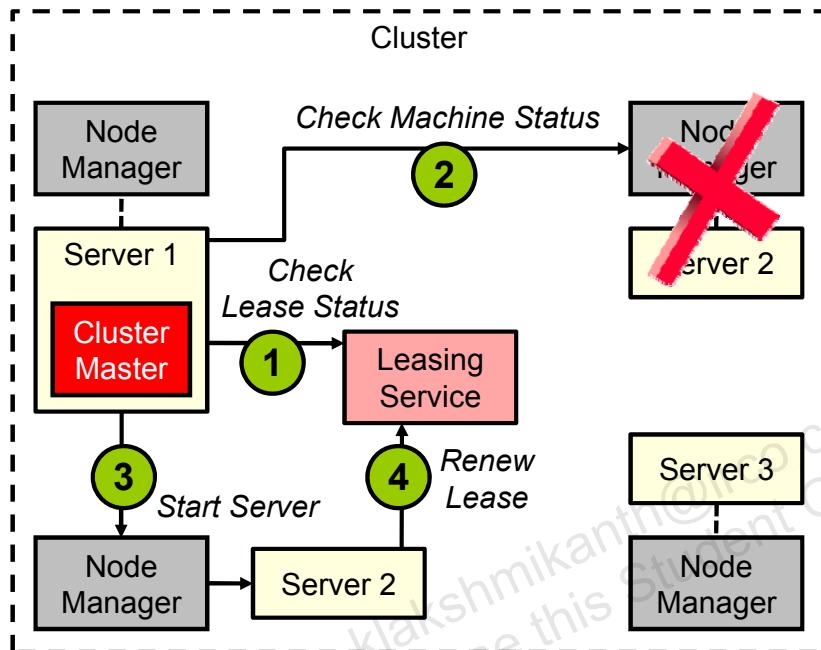
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Automatic Server Migration Architecture: No Failure

The example cluster contains three managed servers, all of which are migratable. Each managed server also runs on its own machine. A fourth machine is also available as a backup, in the event that one of the migratable servers fails. Node Manager is running on the backup machine and on each machine with a running migratable server.

All managed servers in the cluster obtain a migratable server lease from the leasing service. They also periodically renew their leases in the lease table, proving their health and liveness. Because Server 1 starts up first, it also obtains a cluster master lease, whose responsibilities include monitoring the lease table.

## Automatic Server Migration Architecture: Machine Failure



ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Automatic Server Migration Architecture: Machine Failure

1. The machine that hosts Server 2 fails. Upon its next periodic review of the lease table, the cluster master detects that Server 2's lease has expired.
2. The cluster master tries to contact the Node Manager on the backup machine to restart Server 2, but fails, because the entire machine is unreachable. Alternatively, if Server 2's lease had expired because it was hung, but its machine was still reachable, the cluster master would use the Node Manager to restart Server 2 on the same machine.
3. The cluster master contacts the Node Manager on the backup machine, which is configured as an available host for migratable servers in the cluster. The Node Manager then starts Server 2.
4. Server 2 starts up and contacts the Administration Server to obtain its configuration and finally obtains a migratable server lease.

# Leasing Types

WebLogic Server supports two leasing implementation options.

Type	Description
Consensus (in-memory)	<ul style="list-style-type: none"> <li>Servers renew leases by contacting the master directly.</li> <li>Leasing data is maintained in memory on the master and cached on other servers for failover.</li> <li>Node Manager is still required for automatic <b>service</b> migration.</li> <li>This offers better performance.</li> </ul>
Database	<ul style="list-style-type: none"> <li>Servers record lease information to a high-availability database using a JDBC data source or multi data source (non-XA drivers only).</li> <li>Node Manager is not required for automatic <b>service</b> migration.</li> </ul>



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Leasing Types

Using the database leasing implementation, lease information is maintained within a table in a high-availability database. A high-availability database is required to ensure that leasing information is always available. Each member of the cluster must be able to connect to the database to access leasing information. This method of leasing is useful for customers who already have a high-availability database within their clustered environment. This method also allows you to use leasing functionality for JMS and Java Transaction API (JTA) service migration features, without also being required to use Node Manager. Database connectivity is provided through a JDBC data source. Note, however, that XA data sources are not supported for server migration.

In the nondatabase version of Consensus leasing, WebLogic Server maintains leasing information in-memory. This removes the requirement of having a high-availability database to use features that require leasing. One member of a cluster is chosen as the cluster leader and is responsible for maintaining the leasing information. The cluster leader is chosen based on the length of time that has passed since startup. The managed server that has been running the longest within a cluster is chosen as the cluster leader. Other cluster members communicate with this server to determine leasing information; however, the leasing table is replicated to other nodes of the cluster to provide failover.

# Configuration Overview

1. Create machine definitions for your domain.
2. Configure Node Managers on your machines, including network interface settings.
3. Configure the cluster leasing service.
4. Enable automatic server migration on all or a subset of clustered servers.
5. Configure a list of candidate machines for server migration.



*Most migration  
settings require  
server restart.*

**ORACLE®**

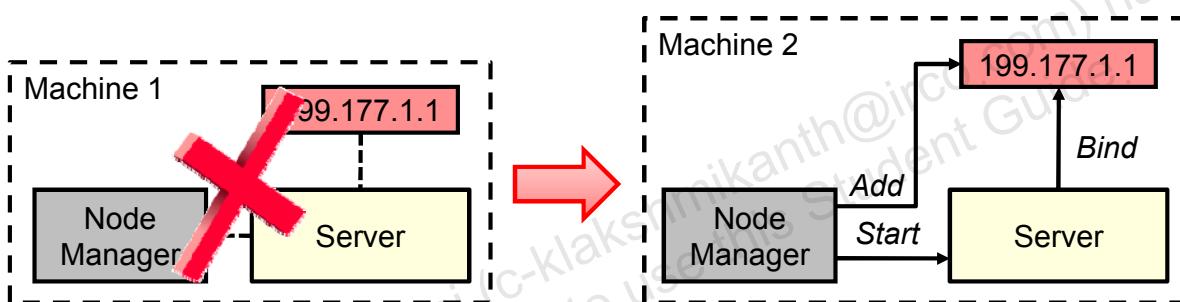
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Configuration Overview

Node Manager must be running and configured to allow server migration. The Java version of Node Manager can be used for server migration on Windows or UNIX. The SSH version of Node Manager can be used for server migration on UNIX only. Refer to the Node Manager Administrator's Guide in the WLS documentation for the available configuration and security options.

## Network Considerations

- To support migration at the network level, a migrated server must continue to use the same static address:
  - Configure your network to supporting floating addresses.
  - Configure each Node Manager to automatically add the address to the new host machine before starting servers.
- Migration is not supported for servers that use network channels to bind to multiple specific addresses.



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

ORACLE

### Network Considerations

If migrated servers do not continue to bind to the same network address, client proxy servers, load balancers, and stub objects (for EJB and JMS clients) will fail to reach the server at its new location.

Each migratable server can be assigned a floating IP address that follows the server from one physical machine to another after migration. The migratable IP address should not be present on the interface of any of the candidate machines before the migratable server is started.

## Node Manager Network Configuration

- Edit the Node Manager configuration and set:
  - The name of the OS network interface
  - The network subnet mask to use for new addresses
- Node Manager delegates the tasks of adding and removing addresses to the wlsifconfig script.

```
nodemanager.properties  
...  
Interface=eth1  
NetMask=255.255.255.0
```



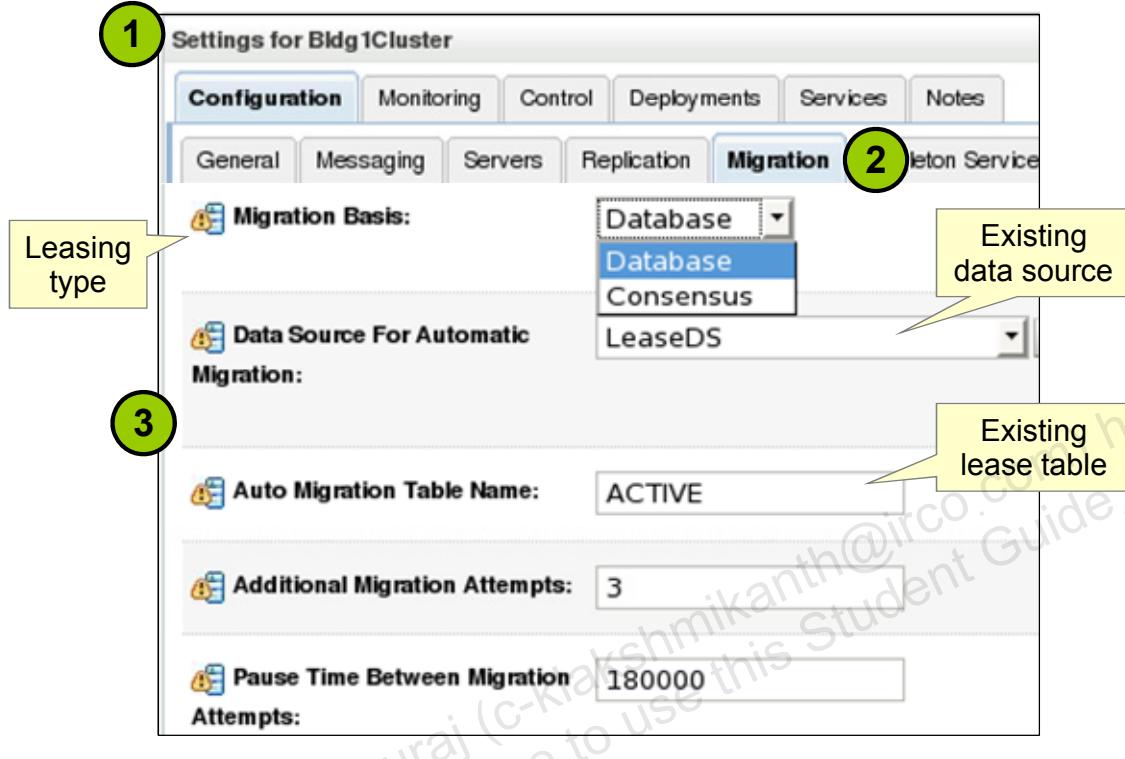
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Node Manager Network Configuration

When using the Java Node Manager, you must edit your nodemanager.properties file and add your environment's Interface and NetMask values. If you are using the SSH version of Node Manager, edit `wlscontrol.sh` and set the `Interface` variable to the name of your network interface.

Both versions of Node Manager use another script, `WL_HOME/common/bin/wlsifconfig.sh/cmd`, to add the address to the host machine before starting the server. It must be able to run the `ifconfig` or `netsh` utilities, which are only available to privileged users. If necessary, you can edit this script so that it is invoked by using `sudo` or a similar command.

# Configuring Cluster Leasing



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

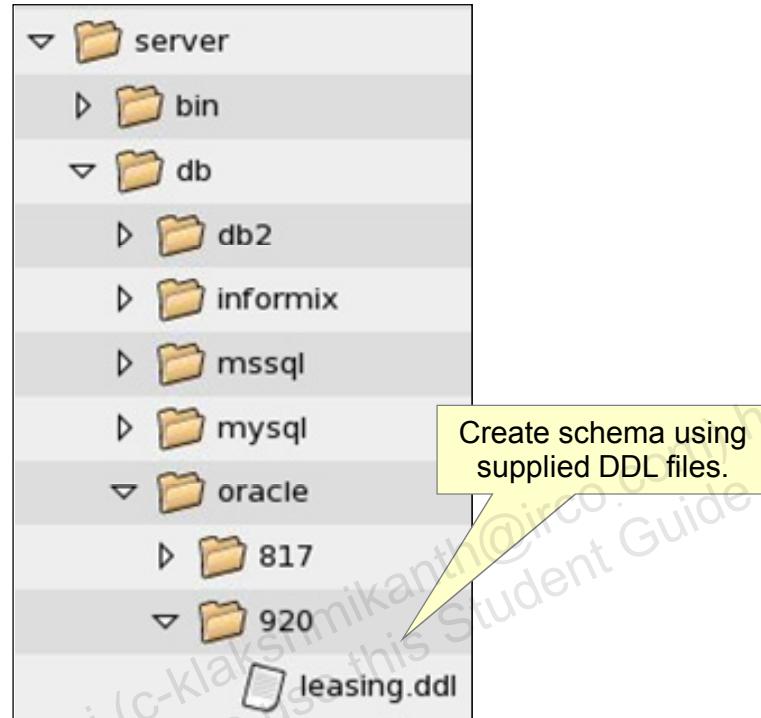
## Configuring Cluster Leasing

Before configuring leasing, ensure that you have created machines and have assigned managed servers to them. Also ensure that you configured and started your Node Managers on each machine.

Shut down all of your clustered servers before modifying the cluster's migration settings. Then perform the following:

1. In the left pane of the console, expand Environment and select Clusters. Then select an existing cluster.
2. Click Configuration > Migration.
3. Select a Migration Basis for the cluster leasing service. For Database leasing, also configure the Data Source For Automatic Migration and Auto Migration Table Name fields . Alternatively, click New to create a new data source. When finished, click Save.

# Database Leasing Schema



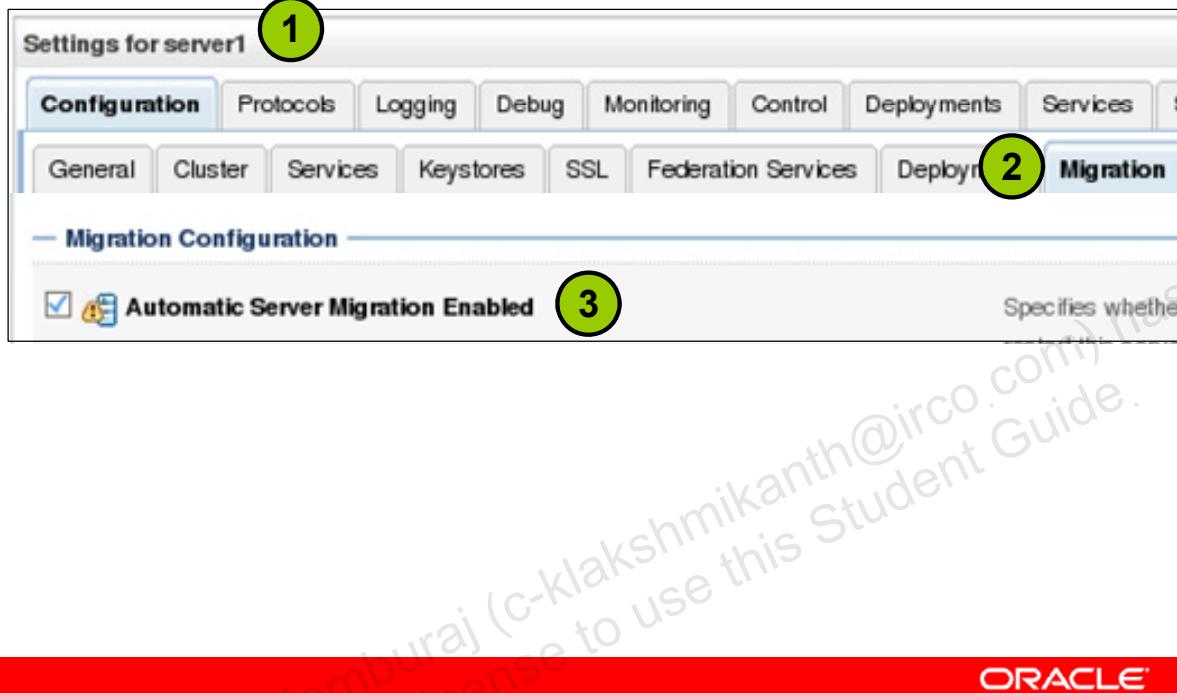
ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Database Leasing Schema

1. Create the leasing table in the database. This is used to store the machine-server associations used to enable server migration. The schema for this table is located in `WL_HOME/server/db/<dbname>/leasing.ddl`, where `dbname` is the name of the database vendor. By default, the database table name is `ACTIVE`, but this can be modified if desired.
2. Set up and configure a data source. This data source should point to the database configured in the previous step.

# Enabling Automatic Migration for a Server



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Enabling Automatic Migration for a Server

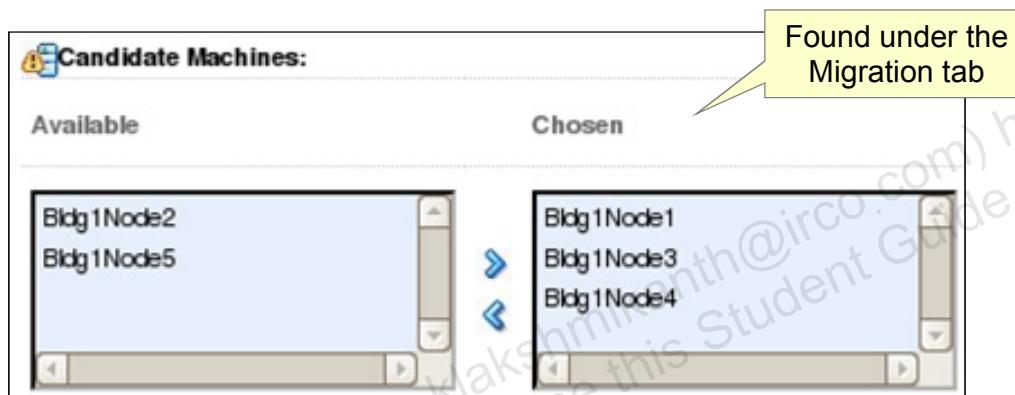
For each server that shall support automatic migration to another machine, perform the following:

1. Select a server in the console.
2. Click Configuration > Migration.
3. Select the check box labeled Automatic Server Migration Enabled and click Save. You must restart the server for this setting to take effect.

## Candidate Machines

Administrators can restrict which machines servers are able to migrate to:

- At the cluster level
- For individual servers (overrides cluster setting)



ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Candidate Machines

To configure the default list of candidate machines for server migration for a cluster:

1. Select a cluster in the console.
2. Click Configuration > Migration.
3. Use the Candidate Machines field to limit the available machines for migration. You can also change the order of preference.

To configure a list of candidate machines that a specific server in the cluster can be migrated to:

1. Select a server in the console.
2. Click Configuration > Migration.
3. Use the Candidate Machines field to limit the available machines for migration. You can also change the order of preference.

## Machine Failback

- When a failed machine becomes available again, WebLogic Server does not automatically restore migrated servers back to it.
- Instead, perform the following:
  1. Gracefully shut down the migrated server using the Node Manager.
  2. Restart the failed machine and Node Manager.



ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Machine Failback

When a machine that previously hosted a server instance that was migrated becomes available again, the reversal of the migration process—migrating the server instance back to its original host machine—is known as failback. WebLogic Server does not automate the process of failback. An administrator can accomplish failback by manually restoring the server instance to its original host.

The general procedures for restoring a server to its original host are as follows:

- Gracefully shut down the new instance of the server.
- After you have restarted the failed machine, restart Node Manager and the managed server.

## Manual Server Migration

- Alternatively, administrators may use the console to manually perform server migration:
  - Gracefully bring down a machine for hardware maintenance.
  - Perform OS or WebLogic Server maintenance.
  - Move a migrated server back to its original machine.
- To help prevent others from making changes during the migration, administrators must first obtain the console lock.



ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

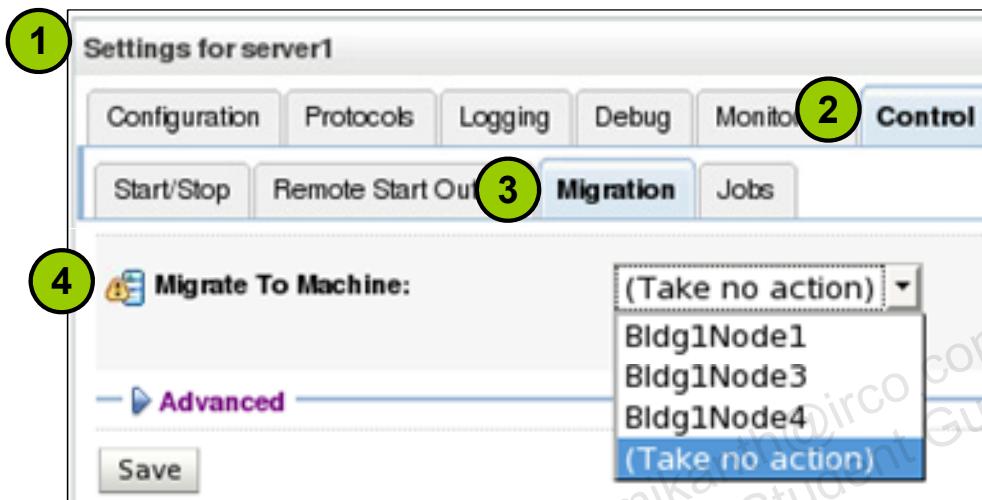
### Manual Server Migration

If automatic server migration is not enabled on any of the servers in a cluster, administrators can still use the console to manually initiate the migration of a server to another candidate machine. However, even if automatic migration is enabled, manual migration may still be desirable to help handle situations in which a server must be brought offline for maintenance.

In this scenario, the following occurs:

1. The administration server contacts the Node Manager on the source machine.
2. The Node Manager on the source machine shuts down the specified server.
3. As part of shutting down, the server removes its lease from the leasing service.
4. The Administration Server contacts the Node Manager on the target machine.
5. The Node Manager on the target machine starts the specified server.

# Migrating a Server



ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Migrating a Server

To manually migrate a server to another machine, perform the following steps:

1. Select a server in the console.
2. Click the Control tab.
3. Click the Migration second-level tab.
4. In the Migrate to Machine field, select a machine from the list of available candidates.  
Then click Save.

## Additional File System Considerations

- To initialize a new machine with the required domain files, use the pack/unpack tools or the nmEnroll WLST command.
- Make all server file stores and transaction logs available to all machines using a highly available shared storage solution.



ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Additional File System Considerations

The nmEnroll command initializes the machine on which WLST is currently running. WLST must be connected to an Administration Server to run this online command. This command downloads and/or generates the minimum required files to run managed servers for the current domain.

To ensure high availability, it is critical that such state information remains available to the server instance and the services it hosts after migration. Otherwise, data about the work in process at the time of failure may be lost. State information maintained by a migratable server, such as the data contained in transaction logs, should be stored in a shared storage system that is accessible to any potential machine to which a failed migratable server might be migrated. For highest reliability, use a shared storage solution that is itself highly available—for example, a storage area network (SAN).

# WebLogic Operations Control (WLOC) Overview

## WLOC:

- Provides an alternative to the standard WLS node manager and server migration features
- Automatically starts clustered Java processes (such as WLS) in response to failures or increasing load
- Prioritizes different types of processes according to defined policies
- Supports a pool of physical and/or virtualized hardware resources
- Employs agent processes to control and observe server processes
- Is managed and monitored from a central Web application



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

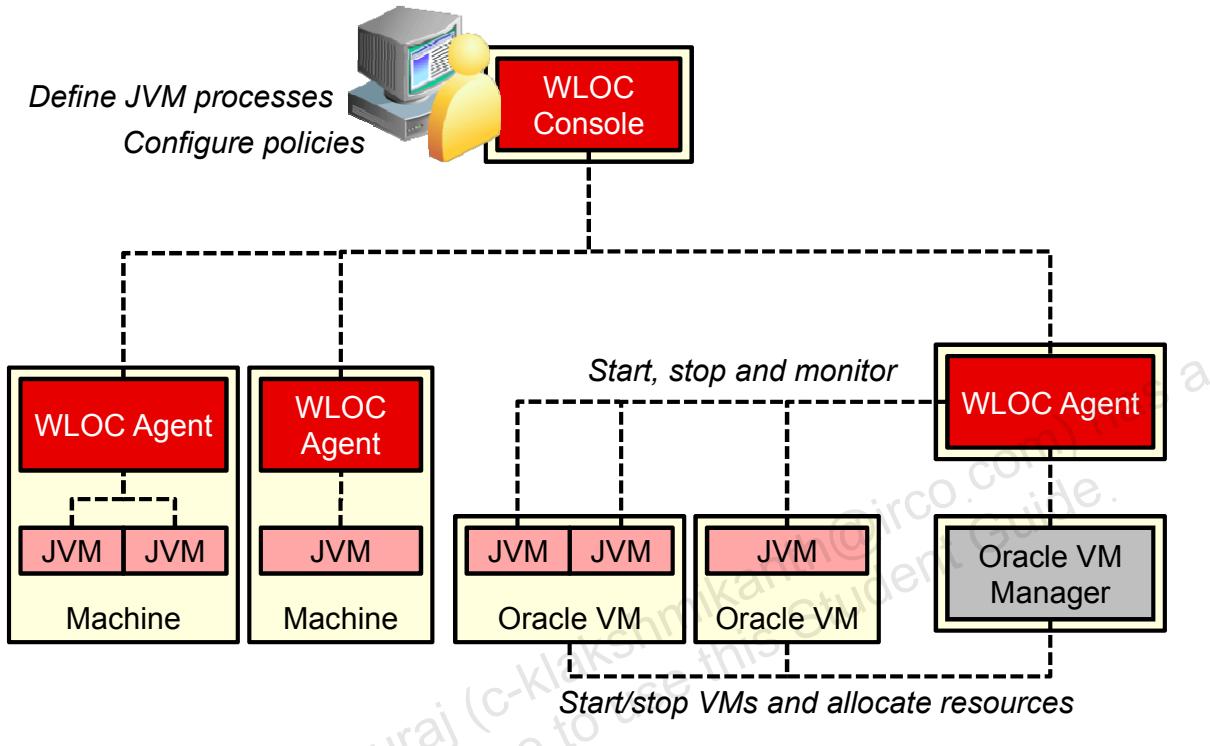
## WebLogic Operations Control (WLOC) Overview

WLOC is a management framework for virtualized and nonvirtualized enterprise Java processes that addresses the key challenges involved in application virtualization. It offers a policy-based framework for creating and automatically enforcing service-level agreements (SLAs) for Java processes like WebLogic Server. WLOC monitors the use of resources across the operations center and distributes the deployment of Java processes in a manner that ensures the most efficient use of available resources. A WLOC resource pool can represent a single physical machine or a collection of virtualized resources that are made available through hypervisor software like Oracle Virtual Machine (VM).

When you deploy a process or when a WLOC action requests that an additional process be started, WLOC examines all resource pools to determine where to host the process. To choose a resource pool, WLOC first eliminates any resource pool that cannot satisfy such dependencies as IP addresses or file systems. For example, if a process requires access to WebLogic Server software, WLOC eliminates any resource pools that do not have WLS installed. After considering declared dependencies, WLOC considers the capacity of each remaining resource pool, the SLAs of any services that are currently deployed, and the relative priorities declared for each service.

When using WLOC in your environment, WLOC agents replace the functionality previously provided by WLS node managers. Both support remote process management and monitoring.

# WLOC Architecture



ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## WLOC Architecture

A WLOC “service” is a collection of one or more processes that WLOC manages as a unit. It includes the details required to run each of the processes. To create SLAs, you define service policies that consist of deployment and runtime constraints along with the actions to take if those constraints are not met.

WLOC agents provide information about the environment to the central WLOC console and controller. Agents start and stop processes and invoke other actions at the request of the controller. A simple agent gathers metrics and manages processes on a single physical machine while a virtual machine (VM) agent gathers data and manages processes across a virtualized resource pool that is supported by one or more physical hosts. For VM agents, this functionality is accomplished through communication with hypervisors such as Oracle VM.

The WLOC controller is the centralized component that gathers data from agents about the operating environment and deployed services to deliver adaptive management. The controller uses the data gathered to intelligently deploy new services and to evaluate and enforce policies to honor the SLAs for all services in the environment.

The controller process also hosts the WLOC administration console, which is a Web browser-based, user interface that you use to configure, manage, and monitor services in your operations center.

# Quiz

Who in a cluster is responsible for initiating an automatic whole server migration?

- a. Node Manager
- b. Database procedure
- c. Cluster Master
- d. Migratable target



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

**Answer: c**

## Quiz

What type of leasing basis does not employ a database?

- a. Exactly-once
- b. WAN
- c. Migratable
- d. Consensus

 ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

**Answer: d**

## Summary

In this lesson, you should have learned how to:

- Describe scenarios in which automatic server migration can be performed
- Explain how automatic server migration works
- Configure consensus or database leasing for a cluster
- Restrict migration to a set of specific candidate machines
- Migrate servers manually to perform routine maintenance
- Compare the WLS server migration framework to WebLogic Operations Control

## Practice 13-1: Migrate Failed Servers

This practice covers the following topics:

- Defining machines and node managers
- Configuring cluster database leasing
- Enabling automatic server migration
- Verifying successful cluster configuration
- Confirming server migration upon machine failure

Unauthorized reproduction or distribution prohibited. Copyright© 2011, Oracle and/or its affiliates.

LakshmiKanth Kemburaj (c-klakshmikanth@irc0.com) has a  
non-transferable license to use this Student Guide.

# 14

## JMS Clustering

ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

# Objectives

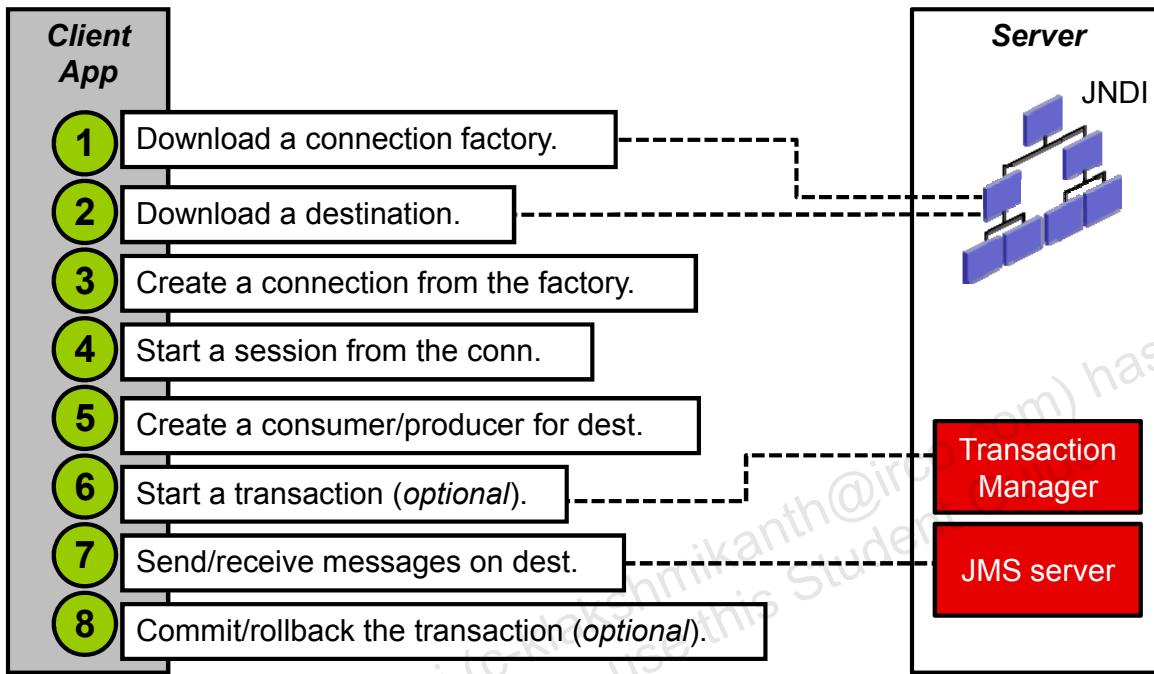
After completing this lesson, you will be able to:

- Describe JMS client communication within a cluster
- Configure automatic Java Message Service (JMS) migration from one server to another
- Use migratable targets for pinned resources
- Explain the load balancing criteria for distributed JMS destinations

# Road Map

- JMS High Availability
  - JMS Communication Review
  - Pinned Resources
  - JMS Cluster Targeting
  - Cluster-Aware JMS Clients
  - Service Migration
  - Candidate Servers
- Distributed JMS Destinations

# JMS Communication Review



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## JMS Communication Review

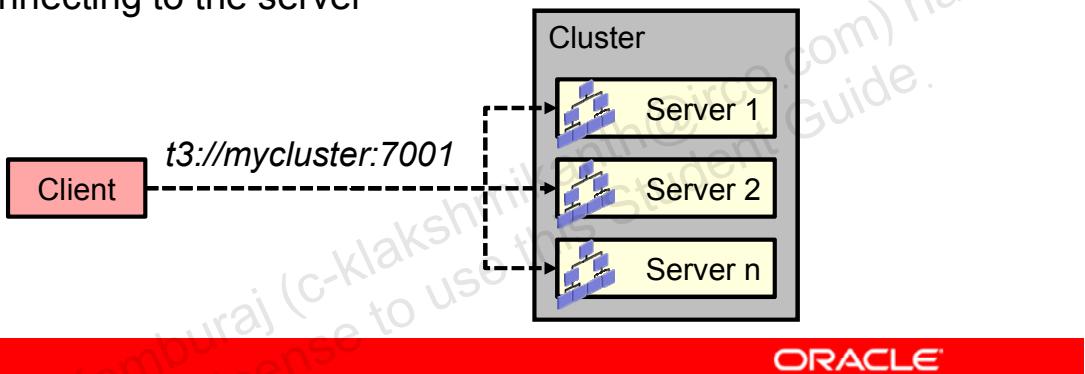
1. WebLogic JMS provides two default connection factories that are included as part of any server's configuration, but administrators can define their own as well. After the connection factory has been defined, you can look it up by first establishing a JNDI context with a local or remote server.
2. Look up a destination (queue or topic) by reusing the same JNDI context in the previous step.
3. Use the connection factory to establish a JMS connection to the server. Depending on the type of connection factory used, this connection may or may not support distributed XA transactions. However, all JMS connections support local (JMS-only) transactions.
4. Create one or more sessions for accessing a queue or topic, using the connection. A session and its message producers and consumers can only be accessed by one thread at a time. Sessions can be transactional or nontransactional, and for consumers support various acknowledgment modes. For transactional sessions, messages are acknowledged when the transaction is committed.
5. Create message producers and/or message consumers for the given destination. Producers and consumers are also associated with an existing JMS session. Message consumers can receive messages synchronously (polling) or asynchronously (notifications).

## JMS Communication Review (continued)

6. JMS clients can participate in a distributed or local transaction.
7. Message producers can send one or more messages to the corresponding destination. Message consumers can receive one or more messages.
8. For producers, a transaction begins and some operations, such as sending messages, are performed. If the transaction commits, all the messages are sent to the destination. If the transaction rolls back, none of the messages arrive at the destination. For consumers, if the transaction commits, the processed messages are acknowledged and removed from the destination. If the transaction rolls back, the messages stay in the destination.
9. When finished, the previously created objects should be closed as soon as possible to free up client and server resources.

## Clustered JNDI Access

- Remote JNDI clients provide a server URL to download objects and stubs.
- To support clusters, WebLogic's JNDI implementation:
  - Supports an initial list of server locations to try
  - Can look up the server list given a DNS name (“cluster address”)
  - Updates the server list with the latest locations upon connecting to the server



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Clustered JNDI Access

When accessing a cluster of servers, a JNDI client does not need to name a specific WebLogic Server to provide its naming service. In fact, it is best for the client to simply request that a WebLogic Cluster provide a naming service, in which case the context factory in WebLogic Server can choose whichever WebLogic Server in the cluster seems most appropriate for the client.

A naming service provider is chosen within WebLogic Server using a cluster's address attribute. This attribute defines the address to be used by clients to connect to a cluster. This address may be either a DNS host name that maps to multiple IP addresses or a comma-separated list of single address host names or IP addresses (for example, t3://node1:7001,node2:7002,node3:7003). If network channels are configured, it is possible to set the cluster address on a per channel basis.

The context that is returned to a client of clustered services is, in general, implemented as a failover stub that can transparently change the naming service provider if a failure (such as a communication failure) with the selected WebLogic Server occurs. When you use a comma-delimited list of DNS names for WebLogic Server nodes, failover is accomplished using the round-robin method, with the request going to a randomly chosen server until that server fails to respond, after which the request will go to the next server on the list. This continues for each server that fails.

## Pinned Resources

- Applications and data sources can be replicated across all servers in a cluster.
- Other types of “singleton” resources must be pinned to a specific server in the cluster:
  - JMS servers and their destinations
  - JMS SAF agents and their imported destinations
  - Persistent stores
  - Transaction manager and logs
  - Custom objects
- To provide availability for pinned resources, you can configure either of the following:
  - Whole server migration
  - Service migration



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Pinned Resources

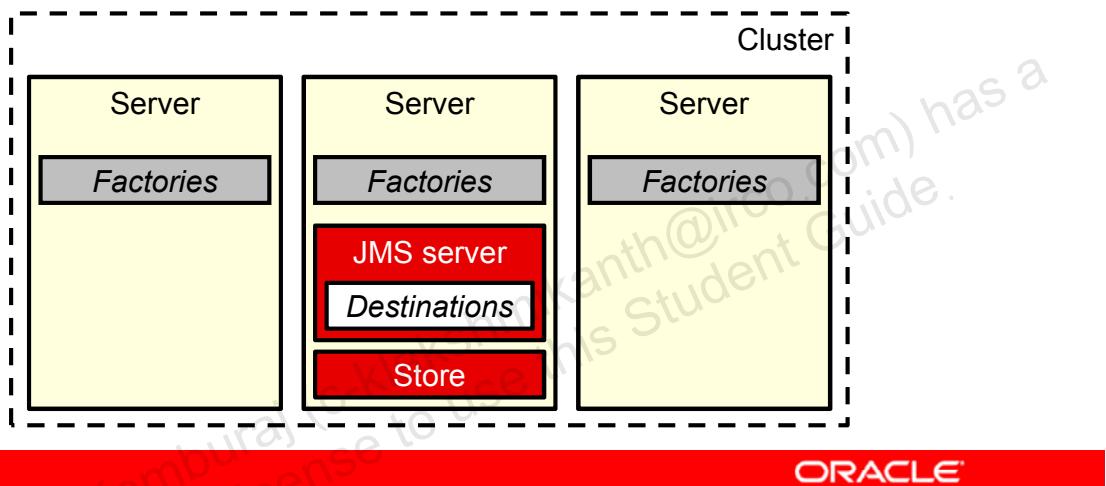
In a WebLogic Server cluster, most subsystem services are hosted homogeneously on all server instances in the cluster, enabling transparent failover from one server to another. In contrast, pinned services, such as JMS-related services and the WebLogic transaction recovery service, are hosted on individual server instances within a cluster. For these services, the WebLogic Server migration framework supports failure recovery with service migration, as opposed to failover. WebLogic Server also supports whole server-level migration, where a migratable server instance and all of its services are migrated to a different physical machine upon failure.

WebLogic Server also provides a framework for migrating custom singleton services as well. You must create a Java class that implements the

`weblogic.cluster.singleton.SingletonService` interface and register it at the domain or application level (`weblogic-application.xml`). When a custom migratable service fails or becomes unavailable for any reason (for example, because of a bug in the service code, server failure, or network failure), it is deactivated at its current location and activated on a new server. The process of migrating these services to another server is handled using the singleton master. The singleton master is a lightweight singleton service that monitors other services that can be migrated automatically. The server that currently hosts the singleton master is responsible for starting and stopping the migration tasks associated with each migratable service.

# JMS Cluster Targeting

- JMS servers and stores are pinned to specific servers.
- Destinations are pinned to specific servers using subdeployments.
- Connection factories are targeted to the entire cluster.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## JMS Cluster Targeting

JMS services are singleton services and, therefore, are not active on all server instances in a cluster. Instead, they are pinned to a single server in the cluster to preserve data consistency. To ensure that singleton JMS services do not introduce a single point of failure for dependent applications in the cluster, WebLogic Server can be configured to automatically or manually migrate them to any server instance in the migratable target list.

An administrator can establish cluster-wide, transparent access to JMS destinations from any server in a cluster, either by using the default connection factories for each server instance in a cluster, or by configuring one or more connection factories and targeting them to one or more server instances in a cluster or to an entire cluster. This way, each connection factory can be deployed on multiple WebLogic servers.

There's a performance advantage to targeting connection factories to exactly the JMS servers/SAF agents that the client will use, as the target set for a connection factory determines the candidate set of host servers for a client connection. Targeting to the JMS servers/SAF agents reduces the likelihood that client connections will connect to servers that do not have a JMS server/SAF agent in cases where there is not a SAF agent on every cluster server. If there is no JMS server/SAF agent on a connection host, the client request must always double-hop the route from the client, to the connection host server, and then ultimately on to the JMS server/SAF agent.

## Clustered Connection Factories

- Similar to EJBs, connection factory “stubs” retrieved by client applications are cluster-aware.
- Within a cluster, connection factories can:
  - Route requests for the specified destination to the actual server hosting that destination
  - Initially try to route requests to the same server that the factory was obtained from
  - Retry failed connections
  - Fail requests over to another server if migration has been performed
- As a best practice, developers should also catch connection errors and in response recreate the connection.

ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Clustered Connection Factories

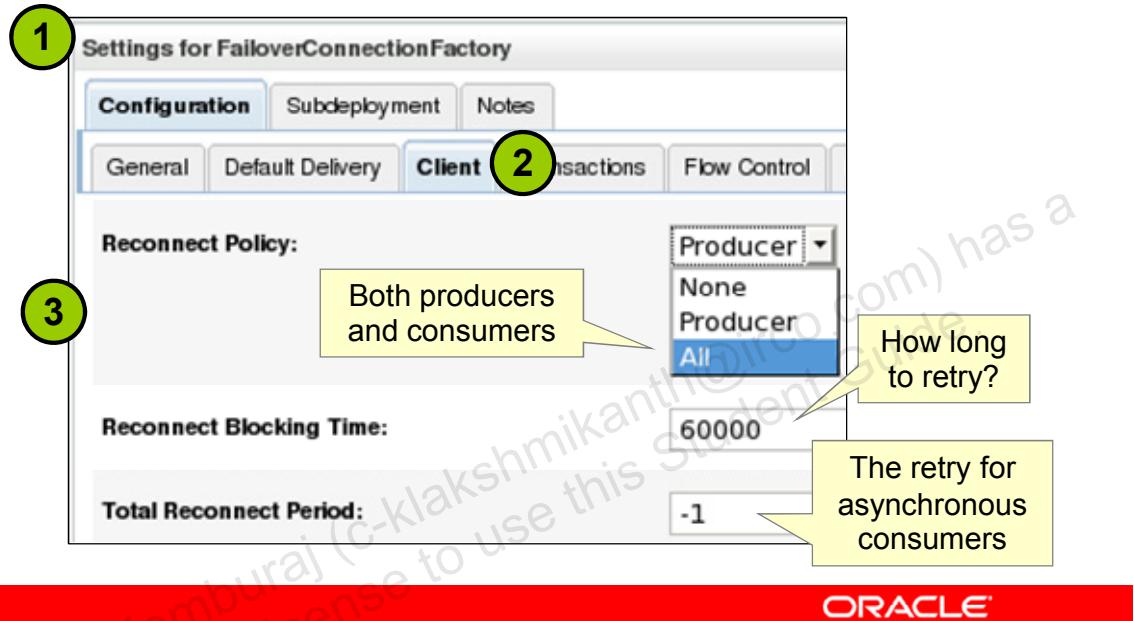
The application uses JNDI to look up a connection factory and create a connection to establish communication with a JMS server. Each JMS server handles requests for a set of destinations. If requests for destinations are sent to a WebLogic Server instance that is hosting a connection factory, but which is not hosting a JMS server or destinations, the requests are forwarded by the connection factory to the appropriate WebLogic Server instance that is hosting the JMS server and destinations.

With the automatic JMS client reconnect feature, if a server or network failure occurs, some JMS client objects will fail over to use another server instance, as long as one is available. For example, if a fatal server failure occurs, JMS clients automatically attempt to reconnect to the server when it becomes available.

WebLogic Server’s automatic reconnection features for JMS clients cannot guarantee transparent failover in every possible failure scenario. Therefore, Oracle recommends that developers continue to programmatically implement reconnection logic in JMS clients. When connection errors are detected, close and recreate the connection using the factory. For asynchronous consumers, developers need to register a custom JMS exception listener object to catch these connection errors.

# Reconnect Policy

By default, only message producers automatically retry failed servers or network connections.



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Reconnect Policy

By default, JMS producer session objects automatically attempt to reconnect to an available server instance without any manual configuration or modifications to existing client code. If you do not want your JMS producers to be automatically reconnected, then you must explicitly disable this feature either programmatically or administratively. In addition, JMS consumer session objects can also automatically attempt to reconnect to an available server, but due to their potentially asynchronous nature, you must explicitly enable this capability.

1. Select an existing connection factory in a JMS module.
2. Click Configuration > Client.
3. Edit the following fields:

**Reconnect Policy:** Select the types of JMS clients that you want implicitly refreshed upon a network disconnect or server reboot. The default value of Producer refreshes only producer-based client objects derived from this connection factory. Select All to refresh all consumer- and producer-based clients derived from this connection factory.

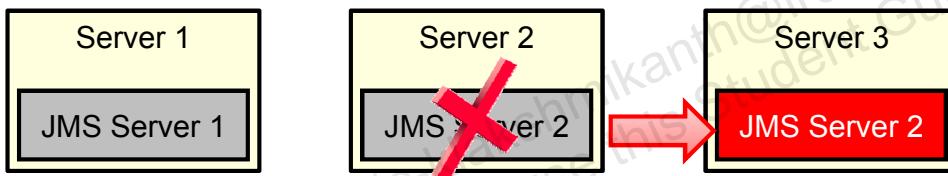
**Reconnect Blocking Time:** Specify how long synchronous JMS calls should block the calling thread before giving up on a JMS client reconnect in progress.

**Total Reconnect Period:** Specify how long asynchronous consumers should keep retrying to reconnect to the server before giving up (-1 = retry indefinitely).

# Service Migration

WebLogic's service migration infrastructure:

- Allows a pinned resource on a failed server to be restarted on another running server
- Requires servers to be members of a cluster
- Supports both manual and automatic failover
- Requires the leasing service for automatic failover
- Cannot be used in conjunction with whole server migration



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Service Migration

Service-level migration in WebLogic Server is the process of moving the pinned services from one server instance to a different available server instance within the cluster. The migration framework provides tools and infrastructure for configuring and migrating targets, and, in the case of automatic service migration, it leverages WebLogic Server's health monitoring subsystem to monitor the health of services hosted by a migratable target.

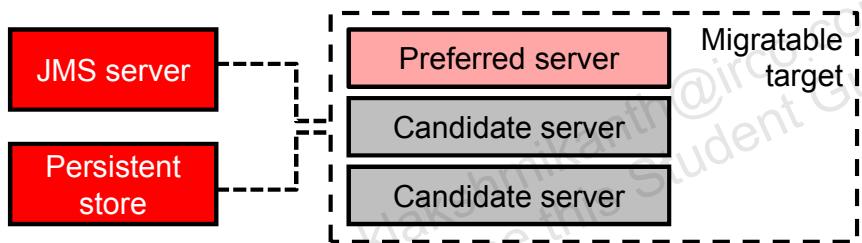
When using consensus leasing, the member servers maintain leasing information in-memory, which removes the requirement of using a database for leasing. But this type of leasing also requires that you use Node Manager to control servers within the cluster. It also requires that all servers that are either migratable or that could host a migratable target must have a Node Manager associated with them. The Node Manager is required to get health monitoring information about the member servers involved.

To accommodate service migration requests, the migratable target performs basic health monitoring on migratable services deployed on it that implement a Health Monitoring Interface. The advantage of having a migratable target do this job is that it is guaranteed to be local. In addition, the migratable target has a direct communication channel to the leasing system and can request that the lease be released (thus triggering a migration) when bad health is detected.

# Migratable Targets

A migratable target:

- Defines a group of servers within a single cluster
- Identifies a primary or “preferred” server
- Can identify a list of candidate servers to use for migration
- Can be assigned as any pinned resource’s target
- Allows you to group resources that should be migrated together



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Migratable Targets

You can configure JMS and JTA services for high availability by using migratable targets. A migratable target is a special target that can migrate from one server in a cluster to another. As such, a migratable target provides a way to group migratable services that should move together. When the migratable target is migrated, all services hosted by that target are migrated.

To configure a JMS service for migration, it must be deployed to a migratable target. A migratable target specifies a set of servers that can host a target and can optionally specify a user-preferred host for the services and an ordered list of candidate backup servers should the preferred server fail. Only one of these servers can host the migratable target at any one time.

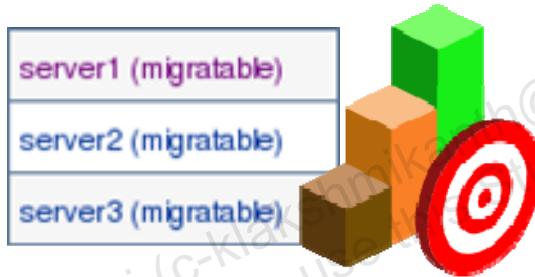
After a service is configured to use a migratable target, the service is independent from the server member that is currently hosting it. For example, if a JMS server with a deployed JMS queue is configured to use a migratable target, then the queue is independent of when a specific server member is available. In other words, the queue is always available when the migratable target is hosted by any server in the cluster.

## Migratable Targets (continued)

A recommended best practice is to limit each migratable target's candidate server set to a primary, secondary, and perhaps tertiary server. Then as each server boots, the migratable targets will be restricted to their candidates rather than being satisfied by the first server to come online. Administrators can then manually migrate services to idle servers.

## Default Migratable Targets

- For convenience, when you add a server to a cluster, a generic migratable target is automatically created that:
  - Has the same name as the server
  - Is configured to use the server as the primary
- If no candidate list is configured, services can be migrated to any server in the same cluster.



ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Default Migratable Targets

The Migratable Target Summary table in Administration Console displays the system-generated migratable targets of “server<name> (migratable),” which are automatically generated for each running server in a cluster. However, these are only generic templates and still must be targeted and configured for automatic migration.

When using migratable targets, you must target your JMS service to the same migratable target used by the custom persistent store. In the event that no custom store is specified for a JMS service that uses a migratable target, then a validation message will be generated by the Administration Server, followed by failed JMS server deployment. Moreover, a server configured in such a way will not boot.

In addition, if the custom store is not targeted to the same migratable target as the dependent service, such as a JMS server, similar validation messages will be generated.

# Service Migration Policies for Migratable Targets

Policy Type	Description
<b>Manual (default)</b>	Automatic service migration is disabled for this target.
<b>Failure recovery</b>	Pinned services deployed to this target will: <ul style="list-style-type: none"><li>Only initially start on the preferred server</li><li>Migrate only if the cluster master determines that the preferred server has failed</li></ul>
<b>Exactly-once</b>	Pinned services deployed to this target will: <ul style="list-style-type: none"><li>Initially start on a candidate server if the preferred one is unavailable</li><li>Migrate if the host server fails <b>or</b> is gracefully shut down</li></ul>



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

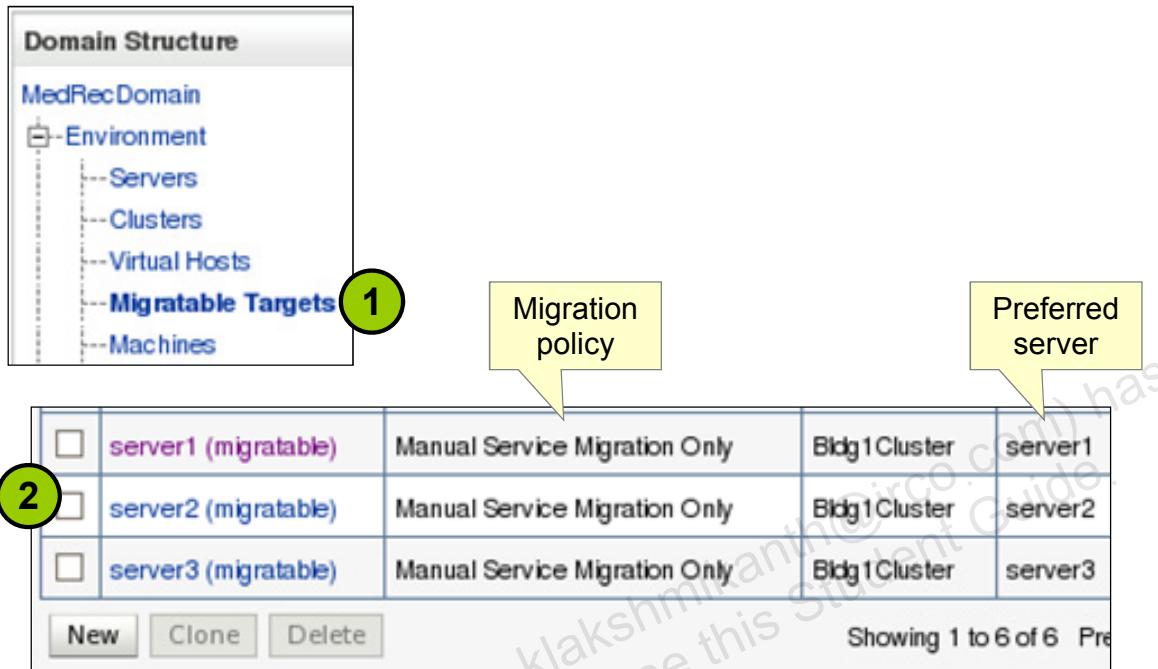
## Service Migration Policies for Migratable Targets

When a migratable target uses the manual policy (the system default), an administrator can manually migrate pinned migratable services from one server instance to another in the cluster, either in response to a server failure or as part of regularly scheduled maintenance.

The failure recovery policy indicates that the service will only start if its preferred server is started. If an administrator manually shuts down the preferred server, either gracefully or forcibly, then services will not migrate. However, if the preferred server fails due to an internal error, then services will be migrated to another candidate server. If such a candidate server is unavailable (due to a manual shutdown or an internal failure), the migration framework will first attempt to reactivate the service on its preferred server. If the preferred server is not available at that time, then the service will be migrated to another candidate server.

The Exactly-once policy indicates that if at least one server in the candidate list is running, then the service will be active somewhere in the cluster if servers fail or are shut down (either gracefully or forcibly). It is important to note that this value can lead to target grouping. For example, if you have five Exactly-once migratable targets and only boot one server in the cluster, all five targets will be activated on that server.

# Configuring Migratable Targets



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

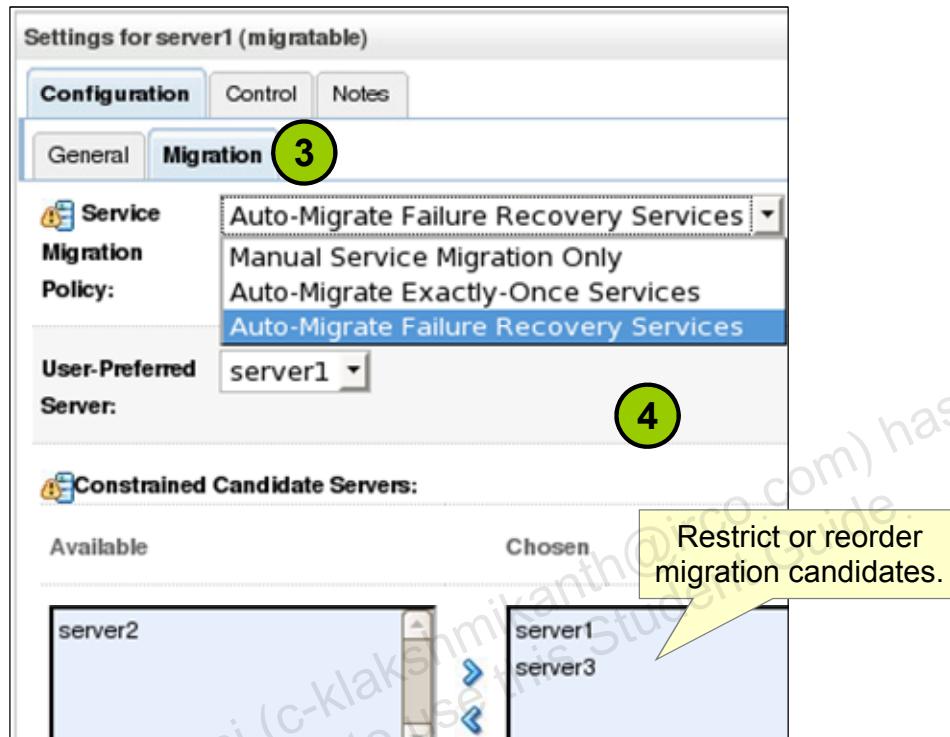
ORACLE

## Configuring Migratable Targets

To customize the policies of an existing migratable target:

1. In the left pane of the console, select Environment > Migratable Targets.
2. Click one of the available migratable targets, or alternatively click New to create a new one.

# Configuring Migratable Targets



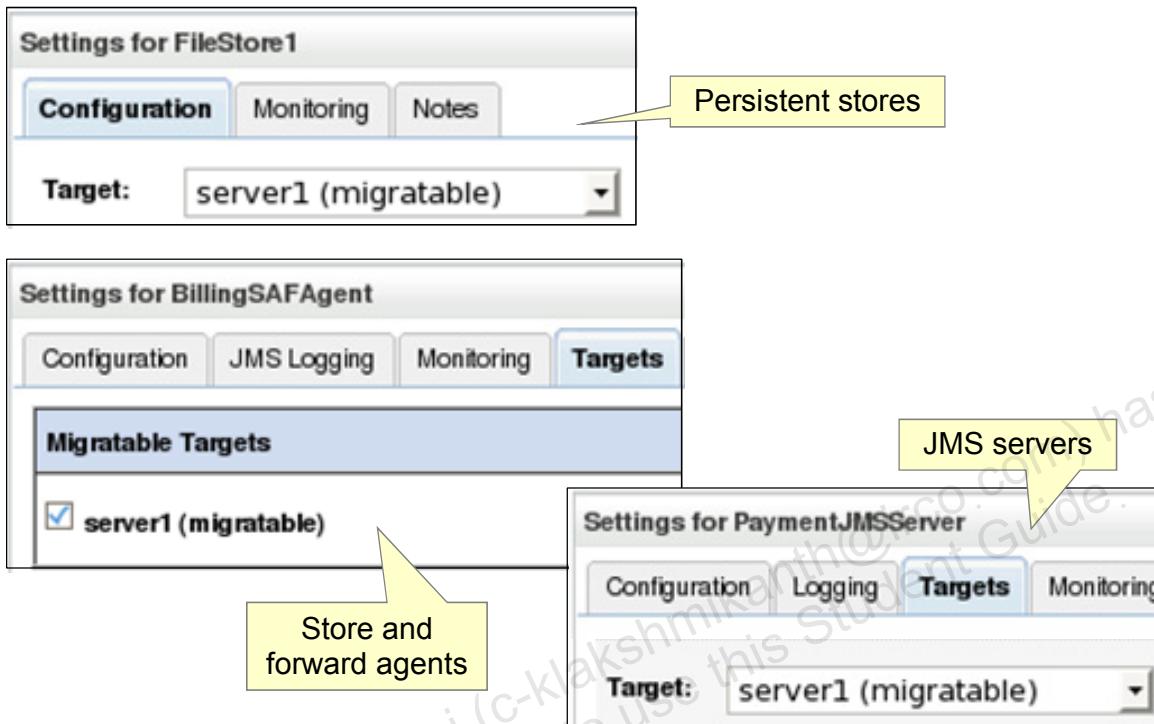
ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Configuring Migratable Targets (continued)

3. Click Configuration > Migration.
4. Edit the following fields and click Save:
  - **Service Migration Policy:** Defines the type of migration policy to use for the services hosted by this migratable target. These options were described earlier.
  - **User-Preferred Server:** Select the server in the cluster that should host targeted services by default.
  - **Constrained Candidate Servers:** Select the servers you want to use as backup destinations for hosted services, and move them to the Chosen list.
  - **Restart On Failure:** Specifies whether or not a failed service will first be deactivated and reactivated in place, instead of being migrated
  - **Seconds Between Restarts:** Specifies how many seconds to wait in between attempts to restart the failed service on the same server
  - **Number Of Restart Attempts:** Specifies how many restart attempts to make before migrating the failed service

# Assigning Resources to Migratable Targets



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Assigning Resources to Migratable Targets

A migratable target is a target that is active on at most one server of a cluster at a time and that controls the servers that can host a pinned migratable service. Migratable services include JMS servers, SAF agents, path service, and custom persistent stores.

You must restart the Administration Server after configuring JMS and other services for automatic service migration. You must also restart any managed servers whose migration policies were modified.

# Automatic Transaction Migration

If you are using WebLogic's distributed transaction manager, you can also configure individual servers to automatically migrate in-progress transactions (**Configuration > Migration**).



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Automatic Transaction Migration

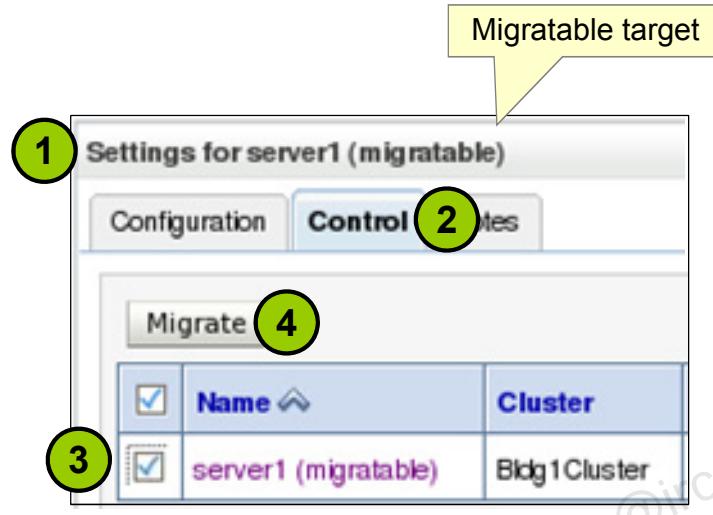
WebLogic's Java Transaction API (JTA) implementation is designed to gracefully handle transaction recovery after a crash. You can specify to have this transaction recovery service automatically migrated from an unhealthy server instance to a healthy server instance with the help of the server health monitoring services. This way, the backup server can complete transaction work for the failed server.

The transaction manager uses the default persistent store to store transaction log files. To enable migration of the transaction recovery service, you must configure the default persistent store so that it stores its data files on a persistent storage solution that is available to other servers in the cluster if the original server fails.

You may also want to restrict the potential servers to which you can migrate the transaction recovery service to those that have access to the current server's transaction log. If no candidate servers are chosen, then any server within the cluster can be chosen as a candidate server.

A server's transaction manager is not assigned to a migratable target like other pinned services are. Instead, manual and automatic JTA is simply configured for each individual server in the cluster. This is due to the fact that the transaction manager has no direct dependencies on other pinned resources when contrasted with services like JMS.

# Migrating Services Manually



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Migrating Services Manually

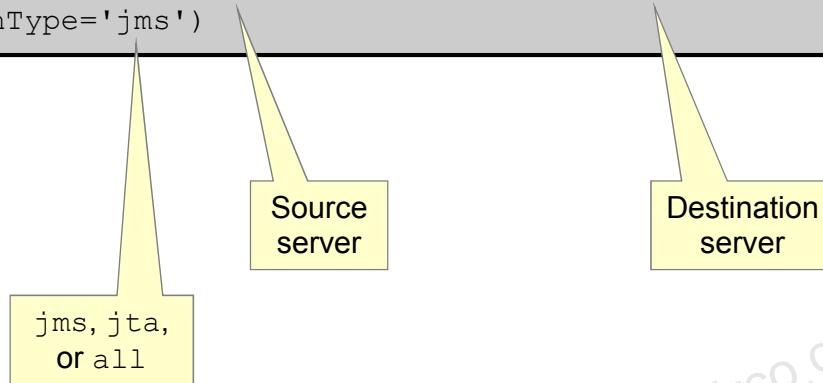
A JMS-related service can be migrated with its hosting migratable target to another server member within a cluster. This includes both scheduled migrations as well as manual migrations in response to a server failure within the cluster. For example, a JMS server, all of its destinations, and its associated persistent store can migrate with their migratable target to a healthy server if the current hosting server should fail. JMS-related services include JMS servers, SAF agents, path service, and custom stores. You can migrate all migratable targets at once or on a target-by-target basis.

1. Lock the console and select an existing migratable target.
2. Click the Control tab.
3. Select the current migratable target.
4. Click Migrate.
5. Select a new server in the cluster to migrate to.
6. Activate your console changes. A request is submitted to migrate the services hosted by this migratable target, and the configuration edit lock is released.

## Service Migration WLST Example

Migrate all JMS related services from one server to another:

```
migrate(sname='server1',destinationName='server3',  
migrationType='jms')
```



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Service Migration WLST Example

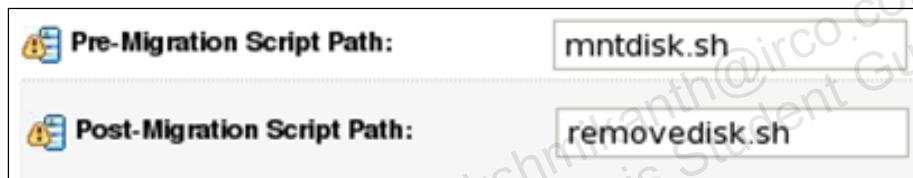
This online command migrates the JMS and/or JTA services of a selected source server to a destination server within the same cluster. The available arguments include:

- **sname:** Name of the server from which the services should be migrated
- **destinationName:** Name of the server to which you want to migrate the services
- **sourceDown:** (optional) Specifies whether the source server is down. This argument defaults to true, indicating that the source server is not running. When migrating JTA services, the sourceDown argument is ignored, if specified, and defaults to true. The source server must always be down in order for the migration of JTA services to succeed.
- **destinationDown:** (optional) Specifies whether the destination server is down. This argument defaults to false, indicating that the destination server is running. If the destination is not running, and you do not set this argument to true, WLST returns a MigrationException. When migrating JMS-related services to a nonrunning server instance, the server instance will activate the JMS services upon the next startup. When migrating the JTA Transaction Recovery Service to a nonrunning server instance, the target server instance will assume recovery services when it is started.

## Pre/Post Migration Scripts

Administrators can register custom script files with migratable targets, which:

- Perform tasks before or after migration (mounting/dismounting a file system, for example)
- Are executed from the domain's `bin/service_migration` folder.
- Also require the Node Manager to be running



**ORACLE**

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Pre/Post Migration Scripts

After creating a migratable target, you may also want to specify whether you are providing any premigration and postmigration scripts. These scripts could be to perform, for example, any unmounting and mounting of a shared custom file store, as needed. The pre/post-migration scripts must be located in the current domain's `/bin/service_migration` directory. A `readme` file is also located here with more detailed information.

Edit a migratable target and locate the following fields:

- **Pre-Migration Script Path:** Before the migratable target is activated, if there is a script specified, and Node Manager is available, then the script will run. Specifying a script without an available Node Manager will result in an error upon migration. If the script fails or cannot be found, migration will not proceed on the current server and will be tried on the next candidate server.
- **Post-Migration Script Path:** After the migratable target is deactivated, if there is a script specified, and Node Manager is available, the script will run.
- **Post-Migration Script Failure Fatal:** Specifies whether or not a failure during execution of the post-deactivation script is fatal to the migration. If it is fatal, the migratable target will not be automatically migrated until an administrator manually migrates it to a server, thus reactivating it.

## Pre/Post Migration Scripts (continued)

- **migrationType:** (optional) Indicate whether to migrate JMS-related services (JMS server, SAF agent, path service, and the WebLogic persistent store) only, or the JTA service only, or all of them. This argument defaults to all.

LakshmiKanth Kemburaj (c-klakshmikanth@irc0.com) has a  
non-transferable license to use this Student Guide.

## Section Summary

In this section, you should have learned how to:

- Explain the deployment constraints for JMS in a cluster
- Describe how remote JMS clients interface with a cluster
- Compare server-scoped and service-scoped migration
- Configure migratable targets and associate them with JMS resources



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Practice 14-1: Configure JMS High Availability

This practice covers the following topics:

- Configuring automatic JMS service migration in a cluster
- Defining and using migratable targets
- Configuring a connection factory for high availability
- Verifying producer and consumer failover after JMS server failure

# Road Map

- JMS High Availability
- Distributed JMS Destinations
  - JMS and Scalability
  - Member Destinations
  - Server Affinity

## JMS and Scalability

- Connection factory and migration features help support JMS high availability.
- Scalability is provided through JMS distributed destinations, which:
  - Can be targeted to multiple JMS servers
  - Can load balance producer/consumer requests across cluster members
  - Can distribute requests uniformly or weighted
  - Avoid failed servers
- Like standard destinations, they:
  - Are bound to JNDI names
  - Support quotas, thresholds, persistence, unit of order, and so on



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

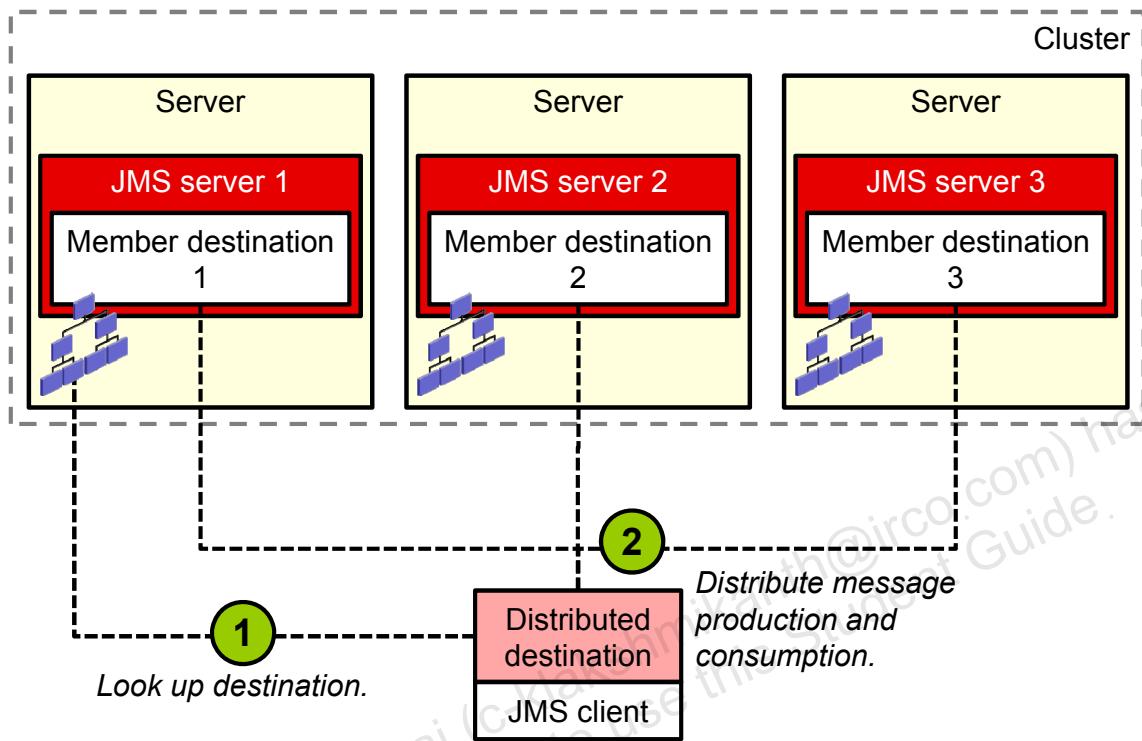
### JMS and Scalability

A distributed destination resource in a JMS module represents a single set of destinations (queues or topics) that is accessible as a single, logical destination to a client (for example, a distributed topic has its own JNDI name). The members of the set are typically distributed across multiple servers within a cluster, with each member belonging to a separate JMS server. Applications that use a distributed destination are more highly available than applications that use stand-alone destinations, because WebLogic JMS provides load balancing and failover for the members of a distributed destination in a cluster.

A uniform distributed destination (UDD) greatly simplifies the management and development of distributed destination applications. Using uniform distributed destinations, administrators do not need to create or designate destination members, but instead rely on WebLogic Server to uniformly create the necessary members on the JMS servers to which a JMS module is targeted. This feature ensures the consistent configuration of all distributed destination parameters, particularly in regards to weighting, security, persistence, paging, and quotas.

The weighted distributed destination feature is still available for users who prefer to manually fine-tune distributed destination members.

# Distributed Destination Architecture



ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Distributed Destination Architecture

Unlike stand-alone queue and topics resources in a module, which can only be targeted to a specific JMS server in a domain, distributed destinations can be targeted to one or more JMS servers, one or more WebLogic Server instances, or to a cluster. For example, targeting a distributed destination to a cluster ensures that a member is uniformly configured on every JMS server in the cluster. You can also use subdeployment groups when configuring distributed destinations to deploy additional JMS resources with the distributed members.

A distributed destination is a set of physical JMS destination members (queues or topics) that is accessed through a single JNDI name. As such, a distributed destination can be looked up using JNDI. The fact that a distributed destination represents multiple physical destinations is transparent to your application.

# Distributed Queues and Topics

Distributed Queues (1-to-1)	
<b>Producers</b>	Perform load balancing across member destinations for each message.
<b>Consumers</b>	Load balance initial consumer creation, but pin to selected member destination.

Distributed Topics (1-to-Many)	
<b>Producers</b>	<ul style="list-style-type: none"> <li>Selected server replicates messages to all member destinations.</li> <li>If a member is unavailable, messages are written to the persistent store and forwarded to it when it becomes available.</li> </ul>
<b>Consumers</b>	Load balance initial consumer creation, but pin to selected member destination.

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Distributed Queues and Topics

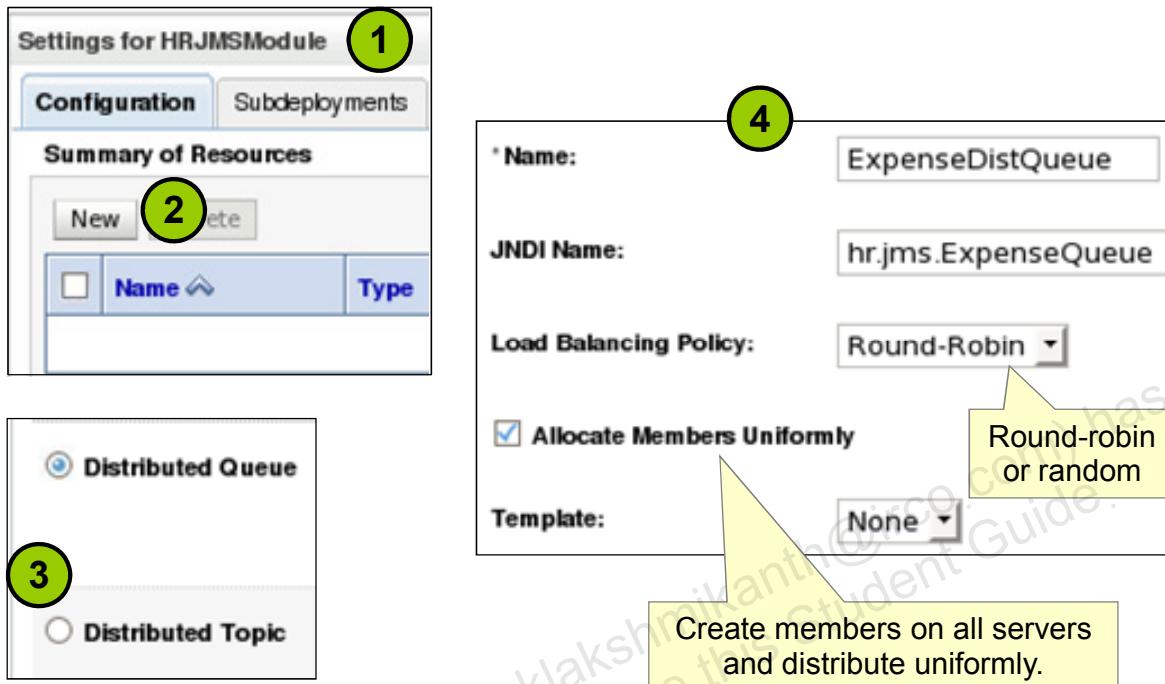
When a message is sent to a distributed queue, it is sent to exactly one of the physical queues in the set of members for the distributed queue. Once the message arrives at the queue member, it is available for receipt by consumers of that queue member only. Each time a message is produced using the sender, a decision is made as to which queue member will receive the message. A single message is not replicated in any way.

When a message is sent to a distributed topic, it is sent to all of the topic members in the distributed topic set. This allows all subscribers to the distributed topic to receive messages published for the distributed topic. If publishers bypass the distributed topic and look up a specific member topic instead, the server still forwards all messages to all members.

If one or more of the distributed topic members is not reachable, and the message being sent is persistent, then the message is stored and forwarded to the other topic members when they become reachable. However, the message can only be persistently stored if the topic member has a JMS store configured.

When creating a JMS consumer object for a distributed queue or topic, a single physical destination member is chosen for the consumer at creation time. The created consumer is then pinned to that member for its lifetime.

# Creating a Distributed Destination



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Creating a Distributed Destination

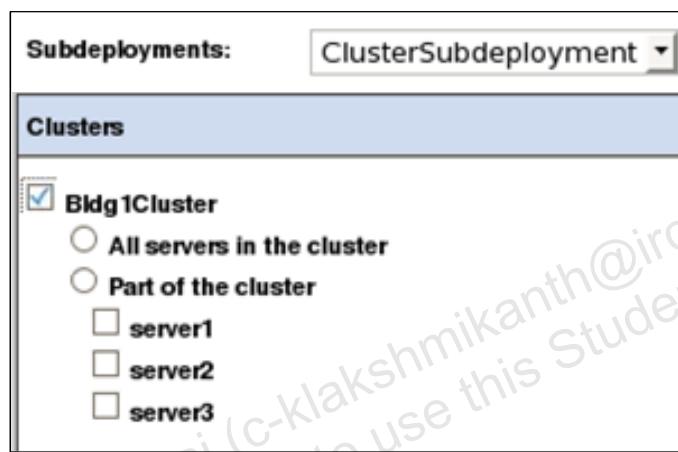
1. Select an existing JMS module.
2. In the Summary of Resources table, click New.
3. Select the Distributed Queue or Distributed Topic option and click Next.
4. For distributed queues, define its basic properties and click Next:
  - **Name:** Enter a name for the uniform distributed queue.
  - **JNDI Name:** Enter the JNDI name used to look up the uniform distributed queue.
  - **Load Balancing Policy:** Specify a policy (Round-Robin or Random) for how messages are distributed to the members of this uniform distributed queue.
  - **Allocate Members Uniformly:** Leave this check box enabled in order for WebLogic Server to uniformly create and allocate the members of this uniform distributed queue on all target JMS servers. If disabled, administrators must manually create, configure, and target member queues.
  - **Template:** Optionally, base this queue on a previously configured destination template.

Administrators can configure additional parameters for the new distributed destination, including thresholds and quotas, producer overrides, logging, and delivery failure options.

## Targeting a Distributed Destination

Use subdeployments to target destinations; either:

- Target specific JMS servers (recommended)
- Target an entire cluster (member destinations will be deployed to all JMS servers running on each server)



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

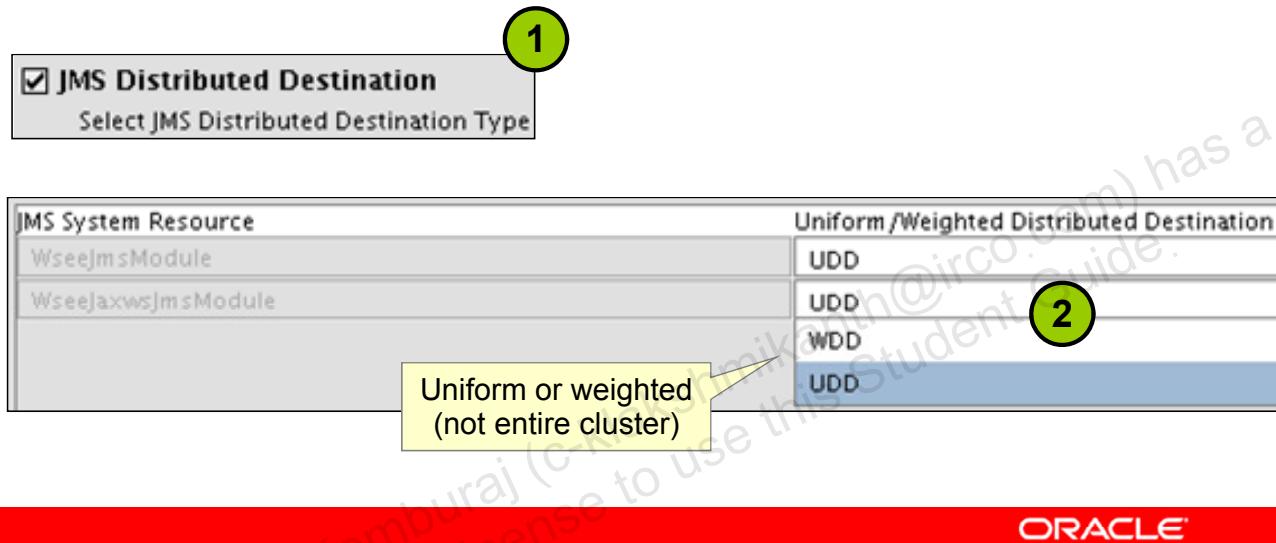
### Targeting a Distributed Destination

When creating a new distributed destination, use the Advanced Targeting option, which allows you to select an existing subdeployment or to create a new one. A subdeployment is a mechanism by which JMS module resources (such as stand-alone destinations, uniform distributed destinations, and connection factories) are grouped and targeted to server resources (such as JMS servers, server instance(s), or a cluster).

For convenience, distributed destinations can be targeted to an entire cluster using a subdeployment. However, use this approach with caution. You may inadvertently deploy destination members to JMS servers that you did not intend to. For example, if a new JMS server is later added to one of the servers in the cluster, the distributed destination will automatically add a new member to this JMS server. Consequently, Oracle recommends targeting distributed destinations to specific JMS servers.

# Distributed Destinations and the Configuration Wizard

If a template includes a JMS module with standard destinations that you are now targeting to a cluster, the wizard can upgrade them to distributed destinations.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Distributed Destinations and the Configuration Wizard

1. In the Select Optional Configuration step of the Configuration Wizard, select the JMS Distributed Destination option and click Next. This option will only be available if one of the source templates contains a JMS module.
2. In the Select JMS Distributed Destination Type step, for each JMS module, you are given the options UDD and WDD. A uniform distributed destination (UDD) will be targeted to an entire cluster. A weighted distributed destination (WDD) will be targeted to a cluster but only activated on specific managed servers. Note that these options only apply if the domain you create contains a cluster. Also note that once this option is set within the Configuration Wizard, it cannot be changed. You must restart the wizard to select different options.

## Distributed Destination WLST Example

Create a uniform distributed queue and target it to a cluster:

```
edit()
startEdit()

jmsModule = getMBean('/JMSSystemResources/HRJMSModule')
sub = jmsModule.createSubDeployment('ClusterSubdeployment')
sub.addTarget(getMBean('/Clusters/Bldg1Cluster'))

moduleResources = getMBean('/JMSSystemResources/HRJMSModule/
    JMSResource/HRJMSModule')
queue = moduleResources.
    createUniformDistributedQueue('ExpenseDistQueue')
queue.setJNDIName('hr.jms.ExpenseQueue')
queue.setSubDeploymentName('ClusterSubdeployment')

save()
activate(block='true')
```



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Distributed Destination WLST Example

Refer to the documentation on the following MBeans:

- JMSSystemResourceMBean
- SubDeploymentMBean (**a component of JMSSystemResourceMBean**)
- JMSBean (**a component of JMSSystemResourceMBean**)
- UniformDistributedQueueBean (**a component of JMSBean**)
- UniformDistributedTopicBean (**a component of JMSBean**)
- DistributedQueueBean (**a component of JMSBean**)
- DistributedTopicBean (**a component of JMSBean**)

# Default Load Balancing Constraints

Policy	Load Balancing Constraint
<b>Server Affinity</b>	A member destination located on the same server is favored over remote members.
<b>Transaction affinity</b>	Messages processed within the same transaction should use the same member destination.
<b>Unit of Order affinity</b>	Messages with the same order ID should use the same member destination.
<b>Number of consumers (queues only)</b>	<ul style="list-style-type: none"> <li><i>Producers</i> favor a member destination with existing consumers over those that have none.</li> <li><i>Consumers</i> favor a member destination that has no other consumers over one that does.</li> </ul>
<b>Store availability (persistent messages only)</b>	A member destination whose JMS server uses a persistent store is favored over one whose JMS server does not.



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Default Load Balancing Constraints

Server affinity implies that JMS clients will first attempt to load balance across any local destination members that are also running on the same WebLogic Server, before load balancing to a remote destination member. This behavior can be disabled if desired.

When producing multiple messages within a transacted session, an effort is made to send all messages produced to the same WebLogic Server. Specifically, if a session sends multiple messages to a single distributed destination, all of the messages are routed to the same physical destination. If a session sends multiple messages to multiple different distributed destinations, an effort is made to choose a set of physical destinations served by the same WebLogic Server.

WebLogic Server directs messages with the same Unit of Order (UOO) to the same distributed destination member. The default routing path to a uniform queue member is chosen by the server based on the hash codes of the UOO name and the uniform distributed queue members. An advantage of this routing mechanism is that routes to a distributed queue member are calculated quickly and do not require persistent storage in a cluster. Alternatively, you can configure a persistent Path Service for your cluster to store UOO routing information within a highly available database.

## **Default Load Balancing Constraints (continued)**

When load balancing consumers across multiple remote physical queues, if one or more of the queues have zero consumers, then those queues alone are considered for balancing the load. Once all the physical queues in the set have at least one consumer, the standard algorithms apply. In addition, when producers are sending messages, queues with zero consumers are not considered for message production, unless all instances of the given queue have zero consumers.

When a persistent message is sent to a distributed destination, or the destination's default delivery mode is persistent, the producer will first attempt to locate a member destination on the same server and that has also been assigned a persistent store. If the local member destination does not have a store, the producer will select a remote member destination with a store.

# Connection Factories and Distributed Destinations

Use custom connection factories to disable load balancing and/or server affinity for distributed destination clients.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Connection Factories and Distributed Destinations

1. Select an existing connection factory.
2. Click Configuration > Load Balance.
3. Edit the available attributes, which pertain to clients of distributed destinations.

The Server Affinity Enabled parameter on connection factories defines whether a WebLogic Server that is load balancing consumers or producers across multiple member destinations in a distributed destination set, will first attempt to load balance across any other local destination members that are also running on the same WebLogic Server.

Applications that use distributed destinations to distribute or balance their producers and consumers across multiple physical destinations, but do not want to make a load balancing decision each time a message is produced, can use a connection factory with the Load Balancing Enabled parameter disabled. To ensure a fair distribution of the messaging load across the members of a distributed destination, the initial physical destination (queue or topic) used by producers is always chosen at random from among the distributed destination members.

# Message-Driven Bean (MDB) Review

## Message-Driven Beans:

- Are standardized JavaEE components
- Are deployed as part of an EJB application
- Automatically consume messages from a specified JMS destination (local or remote)
- Are not invoked directly by clients
- Support various pooling, transaction, and security settings, similar to all EJB types
- Require very little JMS knowledge to implement



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Message-Driven Bean (MDB) Review

An MDB implements loosely coupled, asynchronous business logic in which the response to a request, if any, need not be immediate. It receives messages from a JMS queue or topic, and performs business logic based on the message contents. When a message is received, the EJB container calls the MDB's `onMessage()` method, which contains the business logic the EJB performs.

All instances of a message-driven bean are equivalent—the EJB container can assign a message to any MDB instance. The container can pool these instances to allow streams of messages to be processed concurrently. The EJB container interacts directly with a message-driven bean—creating bean instances and passing JMS messages to those instances as necessary. The container creates bean instances at deployment time, adding and removing instances during operation based on message traffic.

In a topic-based JMS application (the publish/subscribe model), all local instances of an MDB share a JMS session. A given message is distributed to multiple MDBs—one copy to each subscribing MDB. If multiple MDBs are deployed to listen on the same topic, then each MDB receives a copy of every message. A message is processed by one instance of each MDB that listens to the topic.

## MDBs and Distributed Destinations

- MDBs are the preferred type of JMS consumer for WebLogic Server applications.
- When associated with a distributed destination in the same cluster, MDBs:
  - Deploy themselves to every server in the cluster that has a member destination
  - Ensure that every member destination has a consumer
  - Support durable topic subscriptions
- When associated with a remote cluster, MDBs:
  - Transparently fail over to another server after failure or migration
  - Can retry failed connections if failover is not possible



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### MDBs and Distributed Destinations

Collocating an MDB with the destination to which it listens keeps message traffic local and avoids network round trips. Collocation is the best choice if you use WebLogic Server JMS and want to minimize message processing latency and network traffic. Running your MDBs on a separate server or cluster from the destination is suitable if your MDB application is very CPU-intensive. On a separate machine, your application can use 100 percent of the CPU without affecting the operation of the JMS infrastructure.

If you deploy an MDB and the associated distributed destination to the same cluster, WebLogic Server automatically enumerates the distributed destination members and ensures that there is an MDB listening on each member. The MDBs are homogeneously deployed to each clustered server instance. Messages are distributed across the physical destinations on the server instances, and are processed in parallel. If a member of the destination becomes unavailable due to a server failure, message traffic is redirected to the other available physical destinations in the distributed destination set.

In a local cluster, durable subscribers subscribe directly to a distributed destination topic. The MDB deploys only to where there is a distributed destination on the server. In a remote cluster, if durable subscribers subscribe directly to a distributed destination topic, the MDB is deployed once on every distributed destination member.

## Section Summary

In this section, you should have learned how to:

- Discuss the advantages of distributed destinations
- Describe how JMS clients interact with distributed destinations
- Create and target a uniform distributed destination
- Describe how load balancing decisions are made for distributed destination clients



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

# Quiz

Name three types of server resources that can be targeted to a migratable target.

- a. Diagnostic module
- b. Persistent store
- c. Web application
- d. SAF agent
- e. JMS server



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

**Answer:** b, d, e

## Quiz

Which of the following is NOT taken into account when load balancing JMS clients?

- a. Number of consumers
- b. Server affinity
- c. Migratable target
- d. Transaction affinity
- e. No persistent store

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

**Answer: c**

## Lesson Summary

In this lesson, you should have learned how to:

- Describe how JMS servers, destinations, connection factories, and clients function within a cluster
- Automatically restart failed JMS resources on another server
- Use distributed destinations to load balance JMS traffic



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## Practice 14-2: Load Balance JMS Messages

This practice covers the following topics:

- Creating a uniform distributed queue
- Monitoring distributed queue members
- Verifying server affinity for message producers
- Verifying load balancing for message consumers

Unauthorized reproduction or distribution prohibited. Copyright© 2011, Oracle and/or its affiliates.

LakshmiKanth Kemburaj (c-klakshmikanth@irc0.com) has a  
non-transferable license to use this Student Guide.