

Oracle WebLogic Server 11g: Administration Essentials

Volume I—Student Guide

D58682GC20

Edition 2.0

July 2010

D68357

ORACLE®

Authors

Shankar Raman
Steve Friedberg

Technical Contributors and Reviewers

Anand Rudrabatla
Angelika Krupp
Bala Kothandaraman
David Cabelus
Holger Dindler Rasmussen
Matthew Slingsby
Mike Blevins
Mike Lehmann
Nagavalli Pataballa
Serge Moiseev
Shailesh Dwivedi
Steve Button
Takyiu Liu
TJ Palazzolo
Werner Bauer
William Albert
Will Hopkins

Graphic Designer

Satish Bettegowda

Editors

Amitha Narayan
Malavika Jinka

Publishers

Shaik Mahaboob Basha
Veena Narasimhan

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

Preface

1 Introduction

- Objectives 1-2
- Course Prerequisites 1-3
- Course Objectives 1-4
- Course Schedule 1-6
- Facilities in Your Location 1-8
- Summary 1-9

2 Introducing Oracle Fusion Middleware Platform

- Objectives 2-2
- Oracle Fusion Middleware 2-3
- Oracle SOA and Oracle WebCenter Suites 2-5
- Oracle Identity and Access Management 2-6
- Oracle Business Intelligence 2-7
- Oracle Portal, Forms, and Reports 2-8
- Web Tier Components 2-9
- Oracle Fusion Middleware Management Infrastructure 2-10
- Oracle Coherence: In-Memory Data Grid 2-11
- Relationship of Fusion Middleware Products to WebLogic Server 2-12
- Typical Oracle Fusion Middleware Environment 2-13
- Summary 2-14
- Practice 2 Overview: Logging In to the Lab Environment 2-15

3 Defining Java Enterprise Edition Terminology and Architecture

- Objectives 3-2
- Distributed Systems 3-3
- How Standards Help 3-5
- Java EE Standard 3-6
- Java EE Architecture 3-8
- Java Servlets 3-9
- SimplestServlet.java 3-10
- JavaServer Pages (JSP) 3-11
- realsimple.jsp 3-12

Enterprise JavaBeans (EJBs)	3-13
Java Database Connectivity (JDBC)	3-14
Java Naming and Directory Interface (JNDI)	3-15
JNDI Tree	3-16
JNDI Contexts and Subcontexts	3-17
Java Transaction API (JTA)	3-18
Java Message Service (JMS)	3-19
Java Authentication and Authorization (JAAS)	3-20
Java Management Extension (JMX)	3-21
Java EE Connector Architecture (JCA)	3-22
Client Application	3-23
Web Client	3-24
Proxy Server	3-25
Web Server	3-26
Firewalls	3-27
Application Servers	3-28
Web Application Server Configuration	3-29
Application Server Configuration	3-30
Quiz	3-31
Summary	3-32
Practice 3 Overview: Defining Terminology and Architecture	3-33

4 Installing Oracle WebLogic Server 11g

Objectives	4-2
Road Map	4-3
Oracle WebLogic Server Installation	4-4
System Requirements	4-5
GUI Mode Installation	4-6
Choosing or Creating a Home Directory	4-7
Registering for Support	4-8
Choosing an Installation Type and Products	4-9
Choosing the JDK and Product Directory	4-10
Installation and Summary	4-11
QuickStart	4-12
Road Map	4-13
Console and Silent Mode Installations	4-14
Postinstallation: Oracle Home	4-15
Oracle WebLogic Server Directory Structure	4-16
Setting Environment Variables	4-18
Defining Environment Variables	4-19
List of Environment Variables and Their Meanings	4-21

Documentation	4-23
Road Map	4-24
Specialized Installations	4-25
Downloading Software from OTN	4-27
Quiz	4-28
Summary	4-31
Practice 3 Overview: Installing Oracle WebLogic Server 11g	4-32

5 Configuring a Simple Domain

Objectives	5-2
Road Map	5-4
Domain: Overview	5-5
Domain Diagram	5-7
Configuring a Domain	5-8
Starting the Domain Configuration Wizard	5-10
Creating a Domain Using the Domain Configuration Wizard	5-11
Creating a New WebLogic Domain and Selecting the Domain Source	5-12
Configuring Administrator Settings	5-13
Configuring Startup Mode and JDK	5-14
Customizing Optional Configuration	5-15
Configuring the Administration and Managed Servers	5-16
Configuring Clusters and Assigning Servers to Clusters	5-17
Creating an HTTP Proxy Application and Configuring Machines	5-19
Assigning Servers to Machines	5-21
Configuring JDBC Data Sources	5-22
Testing Data Source Connections	5-25
Running Database Scripts	5-26
Configuring the JMS File Store	5-27
Customizing Application and Service Targeting Configuration	5-29
Configuring RDBMS Security Store Database	5-30
Reviewing the WebLogic Domain	5-32
Creating the WebLogic Domain	5-33
Domain Directory Structure	5-34
Road Map	5-36
JVM Run-Time Arguments	5-37
Oracle WebLogic Server Dependencies	5-38
Configuring CLASSPATH	5-39
Starting Oracle WebLogic Administration Server	5-41
Starting Administration Server Using <code>startWebLogic.sh</code>	5-43
Starting the Administration Server by Using the <code>java weblogic.Server</code> Command	5-45

Stopping the Administration Server 5-46
Quiz 5-47
Summary 5-52
Practice 5 Overview: Configuring a Simple Domain 5-53

6 Configuring a Domain Using Templates

Objectives 6-2
Road Map 6-3
Custom Domain Templates 6-4
Domain Template Builder 6-6
Creating a Domain Template 6-7
Comparing Domain Templates 6-8
`wls.jar` Template 6-9
`MedRecTemplate.jar` Template 6-10
Extending an Existing Domain 6-11
Select Which Existing Domain to Extend 6-12
Select What You Want to Extend it With 6-13
Resolving Conflicts 6-14
Summary Before Extending 6-15
Road Map 6-16
Templates for SOA, JDeveloper, and Others 6-17
`oracle.soa_template_11.1.1.jar` Template 6-18
WLS Configuration in the Context of Other Products in the Fusion Middleware Suite 6-20
RCU 6-21
SOA Installation 6-22
Quiz 6-23
Summary 6-26
Practice 6 Overview: Using a Domain Template 6-27

7 Using Administration Console and WLST

Objectives 7-2
Road Map 7-4
Benefits of Using the Administration Console 7-5
Accessing the Administration Console 7-6
Administration Console Login 7-7
Basic Navigation 7-8
Using the Help System 7-9
General Administration Console User Preferences 7-10
Advanced Console Options 7-11
Setting Basic Properties 7-12

Administration Console Monitoring	7-13
Configuration Change Management	7-14
Configuration Change Management Using the Administration Console Change Center	7-15
Domain Configuration Repository	7-16
Configuration Change Process	7-17
Configuration Management Architecture	7-18
XML Schema for config.xml	7-20
Road Map	7-22
WebLogic Scripting Tool (WLST)	7-23
Jython	7-24
Using Jython	7-25
WLST Modes	7-26
WLST Example	7-27
WLST Command Requirements	7-28
Running WLST Scripts	7-29
Importing WLST as a Jython Module	7-30
General WLST Commands	7-31
Offline WLST Commands	7-32
Creating a Domain: Example	7-33
Online WLST Commands	7-34
WebLogic JMX: Overview	7-35
Navigating JMX MBeans	7-36
Generating a WLST Script	7-37
Quiz	7-38
Summary	7-44
Practice 7 Overview: Using the Administrative Console and WLST	7-45

8 Configuring Managed Servers

Objectives	8-2
Road Map	8-3
Configuring Managed Servers	8-4
Creating a Managed Server with WLST	8-5
Starting Oracle WebLogic Managed Servers	8-7
Starting a Managed Server Using startManagedWebLogic.sh	8-8
Command-Line Requirements for Starting the Managed Server Using java weblogic.Server	8-10
Starting a Managed Server Using the Administration Console	8-12
Shutting Down a Server	8-13
Shutting Down a Domain	8-14

Creating a Boot Identity File	8-16
Monitoring All Servers	8-18
Customizing the View for All Servers	8-20
Monitoring Individual Servers	8-21
Demonstration	8-22
Road Map	8-23
Creating a Managed Server on a Remote Computer	8-24
pack and unpack: Examples	8-26
Road Map	8-27
Managed Server Independence (MSI)	8-28
MSI Search Order	8-29
When the Administration Server Is Down	8-31
Running Multiple WLS Instances	8-32
Quiz	8-33
Summary	8-38
Practice 8 Overview: Configuring a Managed Server	8-39

9 Configuring Node Managers

Objectives	9-2
Road Map	9-3
What Node Managers Can Do	9-4
Road Map	9-6
What Is a Machine?	9-7
Relationship of Machines to Other Components	9-8
Creating a Machine	9-9
Defining Names and OS of Machines	9-10
Assigning Servers to a Machine	9-11
Monitoring Machines and Servers	9-12
Configuring a Machine to Use a Node Manager	9-13
Node Manager Architecture	9-14
How a Node Manager Starts an Administration Server	9-15
How a Node Manager Starts a Managed Server	9-16
How a Node Manager Restarts an Administration Server	9-17
How a Node Manager Restarts a Managed Server	9-18
How a Node Manager Shuts Down a Server Instance	9-19
Versions of Node Managers	9-20
Road Map	9-22
Node Manager Default Behaviors	9-23
Configuring a Java-Based Node Manager	9-24
Reconfiguring the Startup Service for a Windows Installation	9-26
Node Manager as a UNIX Daemon	9-27

Reviewing <code>nodemanager.properties</code>	9-28
Configuring a Script-Based Node Manager	9-30
Creating Management OS Users	9-31
Additional Configuration Information	9-32
Configuring the <code>nodemanager.domains</code> File	9-33
Defining the Administration Server Address	9-34
Setting Node Manager Environment Variables	9-35
Node Manager Configuration and Log Files	9-36
Quiz	9-40
Summary	9-43
Practice 9 Overview: Configuring Machines and Node Managers	9-44

10 Viewing and Managing Logs in Oracle WLS Environment

Objectives	10-2
Road Map	10-3
Oracle WebLogic Server Logs	10-4
Server and Domain Logs	10-6
Configuring Server Logging	10-7
Configuring Server Logging: Advanced	10-8
HTTP Access Logs	10-10
Apache Commons Logging API	10-11
Using the Console to View Logs	10-12
Using WLST to View Logs	10-13
Message Attributes	10-14
Message Severity	10-15
Message Catalog Using the Web	10-16
Message Catalog Cross-Reference	10-17
Road Map	10-18
Log Filters	10-19
Creating a Log Filter	10-20
Applying a Log Filter	10-21
Using the Console to Monitor	10-22
Monitoring Running Servers	10-23
Customizing Views	10-24
Monitoring Individual Servers	10-25
Network-Addressing Environment	10-26
Network-Addressing Features	10-27
Quiz	10-28
Summary	10-29
Practice 10 Overview: Viewing and Managing WLS Logs	10-30

11 Deployment Concepts

Objectives	11-2
Road Map	11-3
Overview of Deployment	11-4
What Is Deployed?	11-5
Deployment Process	11-7
Deployment Methods	11-8
Deployment Tools	11-9
Console Deployment Method	11-10
Console Deployment Production Mode	11-11
Preparing a New Application	11-12
Preparing a New Application: Targeting	11-13
Preparing a New Application: Settings	11-14
Deploying or Undeploying Applications	11-15
Redeploying an Application	11-16
Starting and Stopping an Application	11-17
Editing Deployment Descriptors	11-18
Monitoring an Application	11-19
Application Testing	11-20
Deleting Applications	11-21
Command-Line Deployment	11-22
Deployment with <code>weblogic.Deployer</code>	11-23
More <code>weblogic.Deployer</code> Examples	11-24
Deploying Applications with WLST	11-25
Deploying an Application with WLST	11-26
Deployment with WLST	11-27
Road Map	11-28
Production Mode Flag	11-29
Autodeployment	11-30
Autodeploying Using an Expanded Directory	11-31
FastSwap and On-Demand Deployment	11-32
Road Map	11-34
Role of Web Servers	11-35
A Typical Web Interaction	11-36
MIME Types	11-38
HTTP Status Codes	11-39
Static Content	11-40
Dynamic Content	11-41
Configuring Oracle HTTP Server to Serve Multiple WebLogic Servers	11-42
<code>mod_wl_ohs.conf</code>	11-43

Verifying Ports Used by OHS 11-44
Quiz 11-45
Summary 11-48

12 Deploying Java EE Applications

Objectives 12-2
Road Map 12-3
Java EE Web Applications 12-4
Packaging Web Applications 12-6
Web Application Structure 12-7
Web Application Archive 12-8
Optional Configuration of Web Applications 12-9
`web.xml` 12-10
`weblogic.xml` 12-11
`weblogic.xml` Deployment Descriptor 12-12
URLs and Web Applications 12-13
Web Service Applications 12-14
Virtual Directory Mappings 12-15
Virtual Directory Mapping: Example 12-16
Road Map 12-17
EJB Applications 12-18
Types of EJBs 12-19
EJB Application Structure 12-21
`weblogic-ejb-jar.xml` 12-22
Administrator Tasks with EJBs 12-23
Road Map 12-24
What Is an Enterprise Application? 12-25
Typical Java EE System 12-26
Java EE Enterprise Application 12-27
Why Enterprise Applications? 12-29
Enterprise Application Structure 12-30
`weblogic-application.xml` 12-31
Application Scoping 12-32
EAR Class Libraries 12-33
Java EE Library Support 12-34
WebLogic Java EE Shared Libraries 12-35
Shared Library References 12-36
Quiz 12-37
Summary 12-40
Practice 12 Overview: Web Application Deployment Concepts 12-41

13 Advanced Deployment

Objectives	13-2
Road Map	13-3
What Is a Deployment Plan?	13-4
Configuring an Application for Multiple Deployment Environments	13-5
Sample Deployment Plan	13-7
Creating a Deployment Plan	13-8
Creating a New Deployment Plan	13-10
<code>weblogic.PlanGenerator</code>	13-11
Using the Administration Console to Generate a Deployment Plan	13-12
Modifying and Saving Data to Create a New Plan	13-13
New Deployment Plan Shows Changed Values	13-14
Using an Existing Deployment Plan to Configure an Application	13-15
Using an Existing Deployment Plan	13-17
Directory Structure for Easier Production Deployment	13-18
Generic File-Loading Overrides	13-19
Performing a Sanity Check in Production Without Disruption to the Clients	13-20
Road Map	13-21
Staged Deployment	13-22
Road Map	13-23
Application Availability	13-24
Production Redeployment and Application Versioning	13-25
WebLogic Production Redeployment	13-27
Production Redeployment	13-28
Advantages of Production Redeployment	13-29
Requirements and Restrictions for Production Redeployment	13-30
Redeploying a New Application Version	13-31
Redeploying Versus Distributing	13-32
Distributing a New Version of the Production Application	13-33
Distributing a New Application Version	13-35
Production Redeployment	13-36
Quiz	13-37
Summary	13-40
Practice 13 Overview: Deploying Production Applications	13-41

14 Understanding JDBC and Configuring Data Sources

Objectives	14-2
Road Map	14-3
JDBC Review	14-4
JDBC Data Sources	14-5

Data Source Scope	14-6
Multi-Tier Architecture	14-7
Type 4 Drivers	14-8
WebLogic JDBC Drivers	14-9
Road Map	14-10
What Is a Connection Pool?	14-11
JDBC Connection Pooling	14-12
Benefits of Connection Pools	14-13
Modular Configuration and Deployment of JDBC Resources	14-14
How Data Source Connection Pools Are Used	14-15
Creating a Data Source Using the Administration Console	14-16
Non-XA Configuration	14-17
Data Source Connection Properties	14-18
Test Configuration	14-19
Connection Pool Configuration	14-20
Connection Pool Advanced	14-21
Targeting a Data Source	14-22
Viewing the Server JNDI Tree via the Administration Console	14-23
Listing the JNDI Contents Via WLST	14-24
Demonstration	14-25
JDBC URLs	14-26
Connection Properties	14-27
Specifying Connection Properties	14-28
Road Map	14-29
Monitoring and Testing a Data Source	14-30
Connection Pool Life Cycle	14-31
Fixing an Offline Data Source	14-32
Quiz	14-33
Summary	14-36
Practice 14 Overview: Configuring JDBC Data Sources	14-37

15 Setting Up Java Message Service (JMS) Resources

Objectives	15-2
Road Map	15-3
Message-Oriented Middleware	15-4
Point-To-Point Queue	15-5
Publish/Subscribe Topics	15-6
Oracle WebLogic Server JMS Features	15-7
WebLogic JMS Architecture	15-9
Typical JMS Messaging Process	15-10
Transacted Messaging	15-11

JMS Administrative Tasks	15-12
Oracle WLS JMS Implementation	15-13
Road Map	15-14
Creating a JMS Server	15-15
Configuring a JMS Server	15-16
Targeting a JMS Server to a Managed Server	15-17
JMS Modules	15-18
Creating a JMS Module	15-20
Modular JMS Resource Configuration and Deployment	15-21
Connection Factories	15-22
Creating a Connection Factory	15-24
Configuring a Connection Factory	15-25
Destination	15-26
Queue Destinations	15-27
Topic Destinations	15-28
Creating a Destination (Topic)	15-29
Threshold, Quota, and Paging	15-31
Configuring Thresholds and Quotas	15-32
Road Map	15-33
Durable Subscribers and Subscriptions	15-34
How a Durable Subscription Works	15-35
Configuring a Durable Subscription	15-36
Persistent Messaging	15-37
Creating a JMS Store	15-38
Creating a JDBC Store for JMS	15-39
Creating a JMS JDBC Store	15-40
Assigning a Store to a JMS Server	15-41
Persistent Connection Factory	15-42
Configuring Destination Overrides	15-43
Road Map	15-44
Monitoring JMS Servers	15-45
Monitoring and Managing Destinations (Active Queues and Topics)	15-46
Monitoring Queues	15-47
Managing Messages in a Queue	15-48
Quiz	15-49
Summary	15-51
Practice 15 Overview: Configuring JMS Resources	15-52

16 Introduction to Clustering

Objectives	16-2
Road Map	16-3

What Is a Cluster?	16-4
Benefits of Clustering	16-5
What Can Be Clustered	16-6
Proxy Servers for HTTP Clusters	16-7
High Availability for EJBs	16-8
Road Map	16-9
Selecting a Cluster Architecture	16-10
Cluster Architecture	16-11
Basic Cluster Architecture	16-12
Basic Cluster Architecture: Advantages and Disadvantages	16-13
Multitier Cluster Architecture	16-14
Multitier: Advantages and Disadvantages	16-15
Basic Cluster Proxy Architecture	16-17
Multitier Cluster Proxy Architecture	16-18
Proxy Web Server Plug-In Versus Load Balancer	16-19
Proxy Plug-Ins	16-20
OHS as Proxy Web Server	16-21
Request Flow When Using OHS	16-22
WLS HttpClusterServlet	16-23
Road Map	16-24
Server Communication in a Cluster	16-25
One-To-Many Communications	16-27
Considerations When Using Unicast	16-29
Peer-To-Peer Communications	16-30
Clusterwide JNDI Naming Service	16-31
Name Conflicts and Resolution	16-32
Quiz	16-33
Summary	16-35

17 Configuring a Cluster

Objectives	17-2
Road Map	17-3
Preparing Your Environment	17-4
Hardware	17-5
IP Addresses and Host Names	17-6
Cluster Address	17-7
Road Map	17-8
Methods of Configuring Clusters	17-9
Creating a Cluster by Using the Administration Console	17-10
Setting Cluster Attributes	17-12
Configuring Cluster Communication	17-13

Adding Cluster Members: Option 1	17-14
Adding Cluster Members: Option 2	17-15
Creating a Cluster with the Configuration Wizard	17-16
Creating a Cluster Using the Cluster MBean	17-17
Clusters and WLST	17-18
Synchronizing When Starting Servers in a Cluster	17-19
Configuring OHS as Proxy Server	17-21
Starting and Stopping OHS Manually	17-22
Verifying Access Through OHS	17-23
Successful Access of OHS Splash Page	17-24
Quiz	17-25
Summary	17-26
Practice 17 Overview: Configuring Clusters	17-27

18 Managing Clusters

Objectives	18-2
Road Map	18-3
Deploying Applications to a Cluster	18-4
Two-Phase Deployment	18-5
Considerations for Deploying to Cluster	18-6
Production Redeployment in a Cluster	18-7
Road Map	18-8
HTTP Session Failover	18-9
HTTP Session State Replication	18-10
HTTP Session In-Memory Replication	18-11
In-Memory Replication and Proxy Servers	18-12
In-Memory Replication	18-13
In-Memory Replication: Example	18-14
Requirements for In-Memory Replication	18-15
Configuring In-Memory Replication	18-16
Failover with Load Balancer	18-18
Replication Groups	18-19
Configuring Replication Groups	18-21
Replication Groups for Different Criteria	18-22
HTTP Session Persistence Using JDBC	18-23
Configuring JDBC Persistence	18-25
JDBC Persistent Table Configuration	18-26
HTTP Session Persistence Using Files	18-28
Configuring File Persistence	18-29
HTTP State Management Best Practices	18-31
Road Map	18-32

Clustering EJB Objects: Replica-Aware Stub	18-33
EJB: Server Failure Situations	18-34
Load-Balancing Clustered EJB Objects	18-35
Stateless Session Bean Failover	18-36
Configuring EJB Clustering in Deployment Descriptors	18-37
Configuring EJB Clustering Using the Administration Console	18-38
Configuring Clusterable Stateless Session EJBs	18-39
Stateful Session Beans	18-40
Configuring Clusterable Stateful Session EJBs	18-41
Read/Write Versus Read-Only	18-42
Entity Bean Cluster-Aware Home Stubs	18-43
EJB Best Practices	18-44
Quiz	18-45
Summary	18-49
Practice 18: Overview Managing Clusters	18-50

19 Security Concepts and Configuration

Objectives	19-2
Road Map	19-3
Introduction to Oracle WebLogic Security Service	19-4
Oracle Platform Security Services	19-5
Oracle WLS Security Architecture	19-6
Security Services	19-7
Overview of Security Concepts	19-8
Confidentiality	19-9
Credential Mapping	19-11
Road Map	19-12
Security Realms	19-13
Security Model Options for Applications	19-14
How WLS Resources Are Protected	19-16
Users and Groups	19-17
Configuring New Users	19-18
Groups	19-19
Configuring New Groups	19-20
Configuring Group Memberships	19-21
Embedded LDAP Server	19-22
Configuring an Embedded LDAP	19-23
Road Map	19-25
Security Roles	19-26
Configuring the Global Security Role	19-28
Security Policies	19-29

Policy Conditions	19-30
Protecting Web Applications	19-31
Specifying Protected Web Resources	19-32
Defining Policies and Roles for Other Resources	19-33
Configuring Authentication	19-34
Authentication Examples	19-35
Migrating Security Data	19-36
Exporting the WLS Default Authenticator Provider	19-38
Importing into a Different Domain	19-39
Summary	19-40
Practice 19: Overview Configuring Security for WLS Resources	19-41

20 Protecting Against Attacks

Objectives	20-2
Road Map	20-3
What Is SSL?	20-4
Trust and Identity	20-5
Using an SSL Connection	20-6
Enabling Secure Communication	20-8
Oracle WebLogic Server SSL Requirements	20-10
keytool Utility	20-11
Obtaining a Digital Certificate: keytool Examples	20-12
Configuring Keystores	20-14
Configuring SSL for an Oracle WebLogic Server	20-15
Road Map	20-16
Protecting Against Attacks	20-17
Man-in-the-Middle Attacks	20-18
Man-in-the-Middle: Countermeasures	20-19
Configuring a Hostname Verifier	20-21
Denial of Service Attacks	20-22
Denial of Service Attacks: Countermeasures	20-23
Filtering Network Connections	20-24
Connection Filter	20-25
Excessive Resource Consumption	20-26
Large Buffer Attacks	20-27
Setting the Post Size	20-28
Connection Starvation	20-29
User Lockout	20-31
Configuring User Lockout	20-32
Unlocking Users	20-33
Protecting the Administration Console	20-34

Quiz 20-35
Summary 20-37
Practice 20: Overview Configuring Keystores 20-38

21 Backup and Recovery Operations

Objectives 21-2
Road Map 21-3
Review of Terms and Components 21-4
Homes: Oracle, Middleware, WebLogic 21-6
Understanding Backup and Recovery 21-7
Types of Backups 21-9
Backup Recommendations 21-11
Limitations and Restrictions for Backing Up Data 21-12
Performing a Full Offline Backup 21-13
Backing Up a Domain Configuration 21-15
Backing Up an Instance Home 21-16
Creating a Record of Installations 21-17
Road Map 21-18
Directories to Restore 21-19
Recovery After Disaster 21-20
Recovery of Homes 21-21
Recovery of a Managed Server 21-22
Recovery of the Administration Server Configuration 21-23
Restarting an Administration Server on a New Computer 21-24
Recovery of a Cluster 21-25
Restoring OPMN-Managed Components to a New Computer 21-26
Quiz 21-27
Summary 21-32
Practice 21 Overview: Backing Up and Restoring Configuration and Data 21-33

Index

Glossary

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

Carl McGruder (camcgruder@deloitte.com) has a non-transferable
license to use this Student Guide.

Carl McGruder (camcgruder@deloitte.com) has a non-transferable
license to use this Student Guide.

Preface

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

Carl McGruder (camcgruder@deloitte.com) has a non-transferable
license to use this Student Guide.

Profile

Before You Begin This Course

Before you begin this course, you should be able to

- Issue basic UNIX user-level commands
- Perform UNIX desktop navigation tasks
- Describe basic XML concepts
- Edit basic XML documents
- Describe basic TCP/IP networking client/server concepts

How This Course Is Organized

Oracle WebLogic Server 11g: Administration Essentials Edition 2 (based on version 11gR1 PS2 [11.1.1.3.0] - WLS 10.3.3), is an instructor-led course featuring lectures and hands-on exercises. Online demonstrations and written practice sessions reinforce the concepts and skills that are introduced.

Related Publications

Oracle Publications

Title	Part Number
<i>Oracle Fusion Middleware Online Documentation Library 11g Release 1 (11.1.1.3)</i>	E14571_01
<i>Oracle Fusion Middleware Administrator's Guide 11g Release 1 (11.1.1.3)</i>	E10105-04
<i>Oracle Fusion Middleware Upgrade Planning Guide 11g Release 1 (11.1.1.3)</i>	E10125-04
<i>Oracle Fusion Middleware Administrator's Guide for Oracle Web Cache 11g Release 1 (11.1.1.3)</i>	E10143-03
<i>Oracle Fusion Middleware Administrator's Guide for Oracle HTTP Server 11g Release 1 (11.1.1.3)</i>	E10144-03

Additional Publications

- System release bulletins
- Installation and user's guides
- *read.me* files
- International Oracle User's Group (IOUG) articles
- *Oracle Magazine*

Typographic Conventions

The following two lists explain Oracle University typographical conventions for words that appear within regular text or within code samples.

1. Typographic Conventions for Words Within Regular Text

Convention	Object or Term	Example
Courier New	User input; commands; column, table, and schema names; functions; PL/SQL objects; paths	Use the SELECT command to view information stored in the LAST_NAME column of the EMPLOYEES table. Enter 300. Log in as scott
Initial cap	Triggers; user interface object names, such as button names	Assign a When-Validate-Item trigger to the ORD block. Click the Cancel button.
Italic	Titles of courses and manuals; emphasized words or phrases; placeholders or variables	For more information on the subject see <i>Oracle SQL Reference Manual</i> Do <i>not</i> save changes to the database. Enter <i>hostname</i> , where <i>hostname</i> is the host on which the password is to be changed.
Quotation marks	Lesson or module titles referenced within a course	This subject is covered in Lesson 3, “Working with Objects.”

Typographic Conventions (continued)

2. Typographic Conventions for Words Within Code Samples

Convention	Object or Term	Example
Uppercase	Commands, functions	SELECT employee_id FROM employees;
Lowercase, italic	Syntax variables	CREATE ROLE <i>role</i> ;
Initial cap	Forms triggers	Form module: ORD Trigger level: S_ITEM.QUANTITY item Trigger name: When-Validate-Item . . .
Lowercase	Column names, table names, filenames, PL/SQL objects	. . . OG_ACTIVATE_LAYER (OG_GET_LAYER ('prod_pie_layer')) . . . SELECT last_name FROM employees;
Bold	Text that must be entered by a user	CREATE USER scott IDENTIFIED BY tiger ;

14

Understanding JDBC and Configuring Data Sources

ORACLE

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

Objectives

After completing this lesson, you should be able to:

- Configure JDBC and JDBC data sources
- Configure data source scope
- Contrast two-tier and multi-tier JDBC architecture
- Configure a connection pool
- List the benefits of connection pools
- Describe how data sources are used
- Deploy JDBC resources to a target
- View the server JNDI tree
- Complete a connection pool checklist
- Explain the components of JDBC URLs
- Monitor and test a data source



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

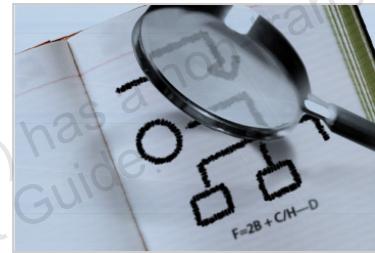
Objectives

Scenario

The Medical Records application needs to store data in a relational database. The application programmers do not have experience with the particular database vendor that you have chosen, but are familiar with SQL from another vendor. They want to isolate the vendor- and platform-specific commands and write generic SQL that would work against any kind of relational database. Eventually, they plan to migrate to Oracle Database, and would like to preserve all of their work now as being vendor agnostic.

Road Map

- Overview of JDBC
 - High-level architecture of JDBC and the driver model
 - Design of a multi-tier architecture
 - Drivers provided by Oracle WebLogic Server
- Data sources
- Monitoring and testing data sources

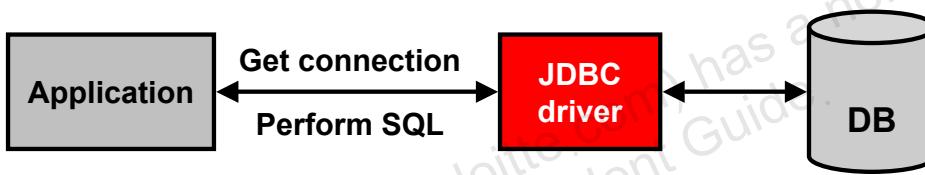


ORACLE

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

JDBC Review

- The Java Database Connectivity (JDBC) specification:
 - Is a platform- and vendor-independent mechanism for accessing and updating a database
 - Provides transparency from proprietary vendor issues
 - Requires the use of a *driver*
- JDBC drivers are supplied by WebLogic Server or by your database vendor.



ORACLE

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

JDBC Review

The JDBC API is a natural Java interface for working with SQL. It builds on Open Database Connectivity (ODBC) rather than starting from the beginning, so programmers familiar with ODBC find it very easy to learn.

The value of JDBC lies in the fact that an application can access virtually any data source and run on any platform with a Java Virtual Machine (JVM). That is, with JDBC, it is not necessary to write one program to access a Sybase database, another to access an Oracle database, another to access an IBM DB2 database, and so on. You can write a single program using the JDBC API. Because the application is written in Java, you need not write different applications to run on different platforms, such as Windows and Linux.

JDBC accomplishes database connections by using a driver mechanism that translates the JDBC calls to native database calls. Although most available drivers are fully written in Java (Type 4) and are thus platform independent, some drivers (Type 2) use native libraries and are targeted to specific platforms.

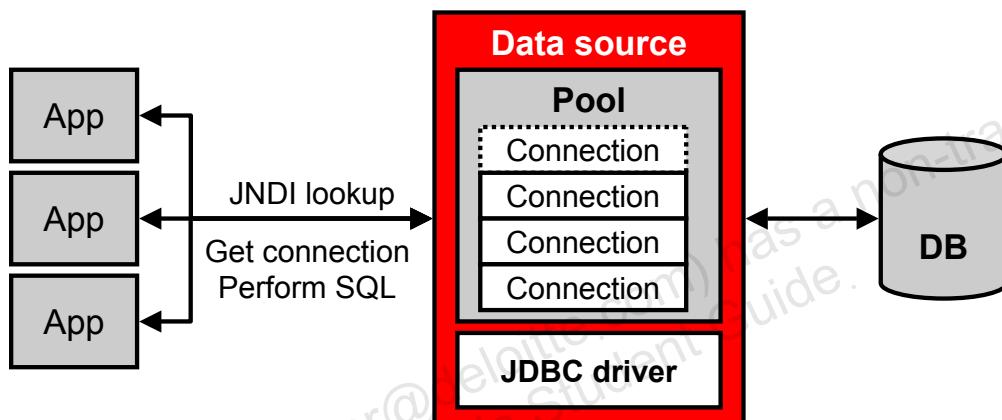
Oracle WebLogic Server includes several Type 4 JDBC drivers, which are compliant with the JDBC 3.0 specification. In addition, the Type 4 drivers support the following JDBC 4.0 specification features:

- Connection validation
- Client information storage and retrieval
- Autoload driver classes (when using Java Platform, Standard Edition 6 [Java SE 6])

JDBC Data Sources

Data sources:

- Enable 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.

JDBC Data Sources

Oracle WebLogic Server can manage your database connectivity through JDBC data sources and multidata 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 the demand, as indicated by the dotted connection at the top of the pool.

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.

Data Source Scope

- Each data source configuration or “module” is persisted as a separate XML document.
- The system modules that are created with the console or WLST are:
 - Stored in the domain’s config/jdbc directory
 - Available to all applications in the domain
- Application-specific modules are:
 - Deployed as part of Java Platform, Enterprise Edition (Java EE) enterprise applications
 - Accessible only by the containing application



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

Data Source Scope

Both Oracle WebLogic Server administrators and developers can define the JDBC data sources. Regardless of which approach you take, each JDBC data source is represented by an XML file that is called a module. The concept of scope is useful when there is a potential namespace clash. For example, if developer 1 makes application A that uses data source X, and developer 2 makes application B that also uses a *different* data source X, the scope is set at an application level. (You may wonder why not name it AX and BX, but that is beside the point.) Alternatively, if both application A and application B wanted to use the *same* data source X, it would be scoped at the server level by the administrator.

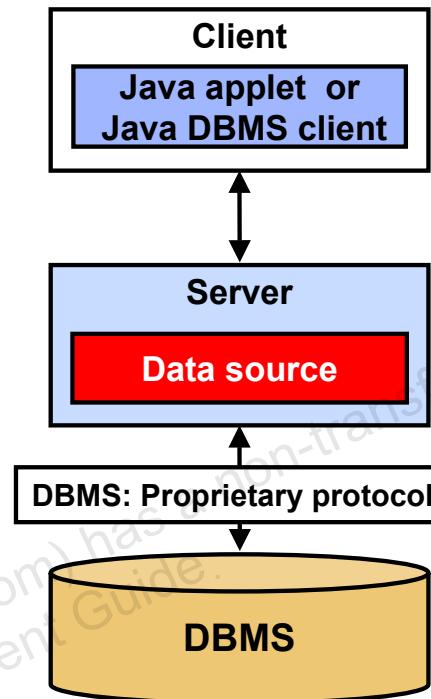
WebLogic administrators typically use the Administration Console or the WebLogic Scripting Tool (WLST) to create and deploy (target) JDBC modules. These JDBC modules are considered system modules, are stored in the domain’s configuration repository as separate XML files, and are referred to by the domain’s config.xml file.

Alternatively, developers define data sources in XML descriptor files, and then package the JDBC modules within a Java EE enterprise application for administrators to deploy. These JDBC modules are considered application modules. Because the modules are deployment descriptors, they can also be modified for different environments using Java EE deployment plans.

All WebLogic JDBC module files must end with the -jdbc.xml suffix, such as examples-demo-jdbc.xml. Oracle WebLogic Server checks the file name when you deploy the module.

Multi-Tier Architecture

- In the multi-tier model, commands are sent to a “middle tier” of services, which then sends the commands to the DBMS.
- The DBMS processes the commands and sends the results back to the middle tier, which then sends them to the client.



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

Multi-Tier Architecture

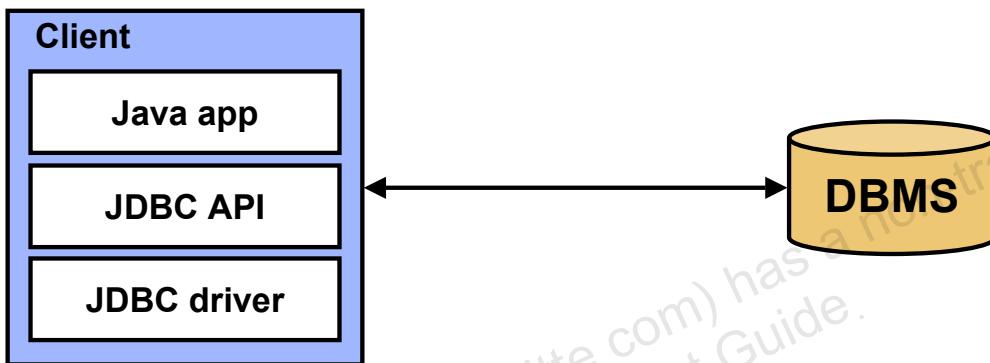
The middle tier makes it possible to maintain control over access and the kinds of updates that can be made to corporate data. Another advantage is that it simplifies the deployment of applications. Finally, in many cases, the multi-tier architecture can provide performance advantages.

Until recently, the middle tier has typically been written in languages such as C or C++, which offer fast performance. However, with the introduction of optimizing compilers that translate Java bytecode into efficient machine-specific code and technologies, such as Enterprise JavaBeans, the Java platform is fast becoming the standard platform for middle-tier development. This is a big plus, making it possible to take advantage of Java's multithreading and security features.

With enterprises increasingly using the Java programming language for writing server code, the JDBC API is being used more and more in the middle tier of a three-tier architecture. Some of the features that make JDBC a server technology are its support for connection pooling, distributed transactions, and disconnected rowsets. The JDBC API is what allows access to a data source from a Java middle tier.

Type 4 Drivers

Type 4 drivers are “all-Java” driver implementations that do not require client-side configuration.



ORACLE

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

Type 4 Drivers

A Type 4 driver is a database driver that is written in 100% pure Java. Drivers that are written in Java have all the performance benefits because they do not have the extra layers between the program and the database. They can operate on any platform and can be downloaded from a server (for example, when using an applet). Because the driver can be downloaded from a server, the client machine does not require preconfiguration of a native driver. This preconfiguration is why the Type 1, 2, and 3 drivers are now deprecated. All that remains are Type 4 drivers.

WebLogic JDBC Drivers

- Oracle and third-party drivers are included in the WLS installation for many popular database products:
 - Oracle 9i, 10g, and 11g
 - Sybase Adaptive Server
 - Microsoft SQL Server
 - IBM DB2
 - Informix
 - MySQL
 - PointBase
- By default, these drivers are added to the server's classpath.

ORACLE

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

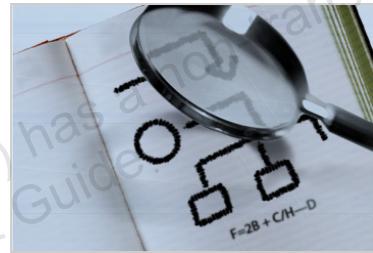
WebLogic JDBC Drivers

The WebLogic Type 4 JDBC drivers are installed with Oracle WebLogic Server in the <WL_HOME>/server/lib folder, where <WL_HOME> is the directory in which you installed Oracle WebLogic Server. Driver class files are included in the manifest classpath in weblogic.jar, so the drivers are automatically added to your classpath on the server.

This release includes support for Oracle 11g and 11g Real Application Clusters (RAC). Support for 11g RAC continues to rely on the well-proven integration architecture using multidata sources for X/Open Distributed Transaction Processing (XA) with load balancing.

Road Map

- Overview of JDBC
- Data sources
 - Describing a data source and how it works
 - Using the Administration Console to create a data source
- Monitoring and testing data sources



ORACLE

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

What Is a Connection Pool?

- A connection pool is a group of ready-to-use database connections associated with a data source.
- Connection pools:
 - Are created at Oracle WebLogic Server startup
 - Can be administered using the Administration Console
 - Can be dynamically resized to accommodate increasing or decreasing load

ORACLE

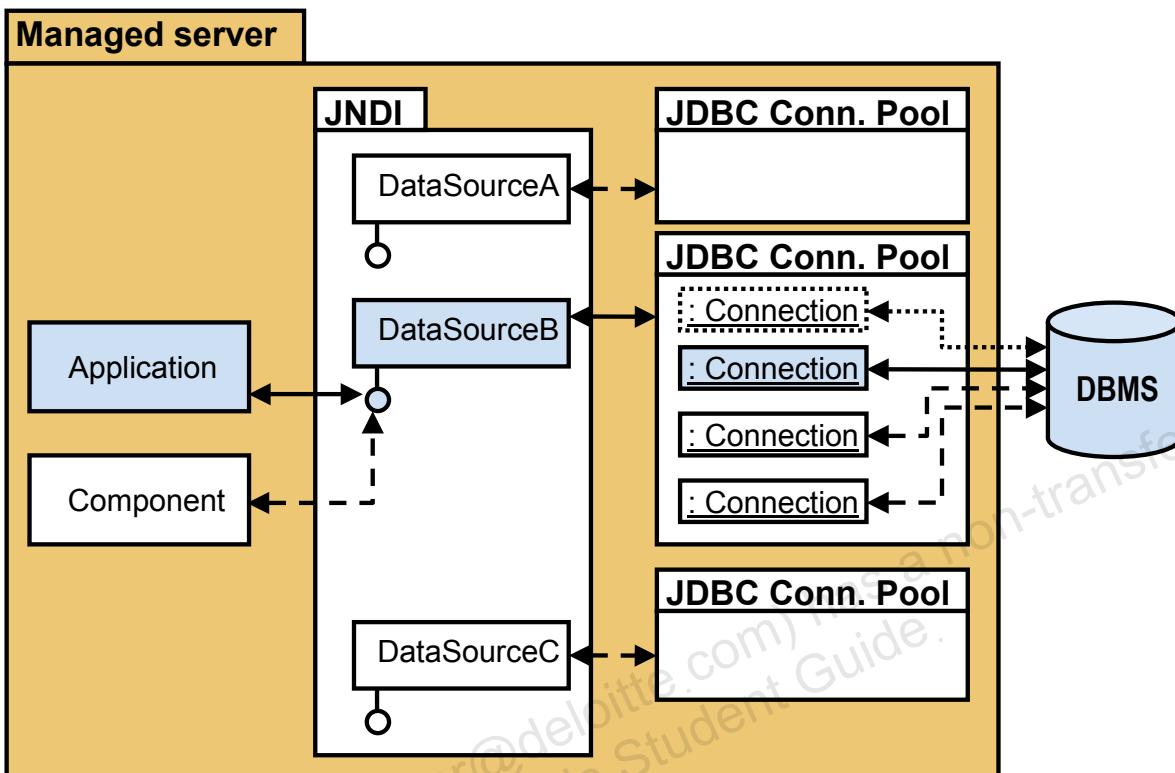
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

What Is a Connection Pool?

Oracle WebLogic Server opens JDBC connections to the database during the WebLogic startup process and adds the connections to the pool. This is faster than creating a new connection on demand. The size of the pool is dynamic and can be fine-tuned.

The connection pool within a JDBC data source contains a group of JDBC connections that applications reserve, use, and then return to the pool. The connection pool and the connections within it are created when the connection pool is registered, usually when starting up Oracle WebLogic Server or when deploying the data source to a new target.

JDBC Connection Pooling



ORACLE

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

JDBC Connection Pooling

The graphic in the slide shows the flow from the applications through the JNDI tree, through the JDBC connection pools, and finally to the database. In Oracle WebLogic Server, you can configure database connectivity by configuring the JDBC data sources and the multi-data sources, and then targeting or deploying the JDBC resources to the servers or clusters in your WebLogic domain.

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. Applications look up a data source on the 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 calls `connection.close()`, which returns the connection to the connection pool in the data source.

Benefits of Connection Pools

- The following are some advantages of connection pooling:
 - Connection time and overhead are saved by using an existing database connection.
 - It facilitates easier management because connection information is managed in one location.
 - The number of connections to a database can be controlled.
 - The DBMS can be changed without the application developer having to modify the underlying code.
- A connection pool allows an application to “borrow” a DBMS connection.

ORACLE

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

Benefits of Connection Pools

Making a DBMS connection is a very slow process when compared to assigning an existing connection. When Oracle WebLogic Server starts, connections from the connection pools are opened and are available to all clients. When a client closes a connection from a connection pool, the connection is returned to the pool and is available for other clients; the connection itself is not closed. There is little cost in opening and closing pool connections. The alternative is for application code to make its own JDBC connections as needed. A DBMS runs faster with dedicated connections than if it has to handle incoming connection attempts at run time.

Connection information, such as the JDBC driver class name, the database location (URL), and the username and password can be managed in one location using the Administration Console. Application developers can obtain a connection without having to worry about these details.

Limiting the number of DBMS connections is important if you have a licensing limitation for DBMS connections or a resource concern.

Clients use a connection pool by “borrowing” a connection, using it, and then returning it to the pool by closing it. The connection pool can grow or shrink dynamically to accommodate demand. The Administration Console is used to set a connection pool’s *initial capacity*, *maximum capacity*, and *capacity increment*.

Modular Configuration and Deployment of JDBC Resources

- The JDBC configurations in WebLogic Server are stored in XML documents:
 - All JDBC configurations must conform to the new `weblogic-jdbc.xsd` schema.
 - IDEs and other tools can validate the JDBC modules based on the schema.
- You create and manage JDBC resources either as system modules or as application modules.
- The JDBC application modules are a WLS-specific extension of Java EE modules and can be deployed either within a Java EE application or as stand-alone modules.



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

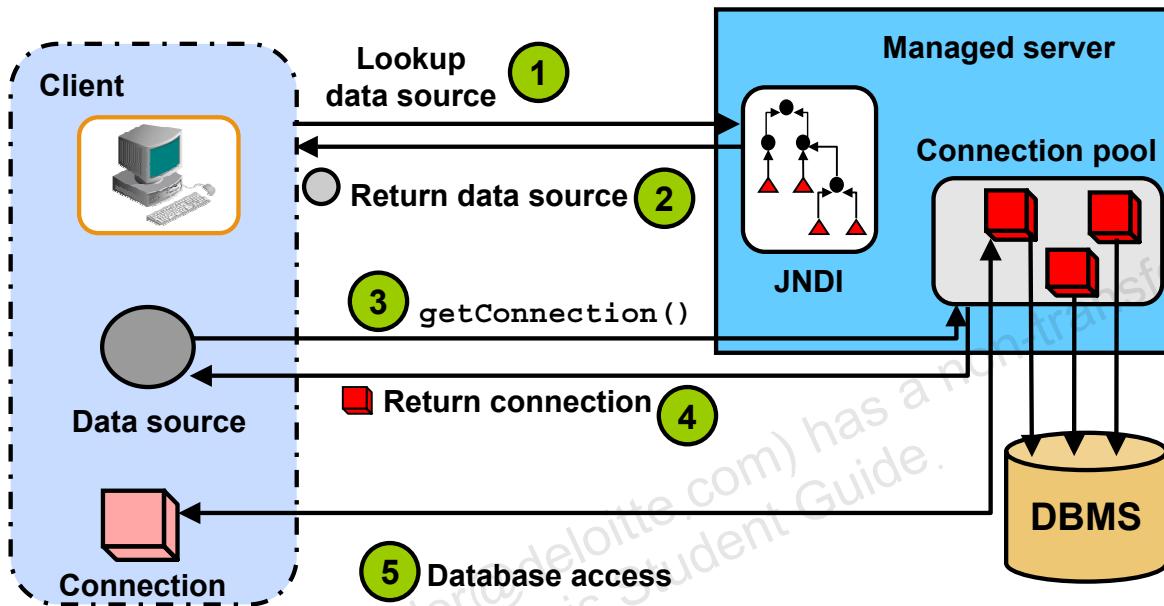
Modular Configuration and Deployment of JDBC Resources

Example of a JDBC configuration:

```
<?xml version="1.0" encoding="UTF-8"?>
<jdbc-data-source xsi:schemaLocation="http://xmlns.oracle.com/weblogic/jdbc-data-
source http://xmlns.oracle.com/weblogic/jdbc-data-source/1.0/jdbc-data-
source.xsd" xmlns="http://xmlns.oracle.com/weblogic/jdbc-data-source"
xmlns:sec="http://xmlns.oracle.com/weblogic/security"
xmlns:wls="http://xmlns.oracle.com/weblogic/security/wls"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <name>MedRecGlobalDataSourceXA</name>
  <jdbc-driver-params>
    <url>jdbc:oracle:thin:@localhost:1521:orcl</url>
    <driver-name>oracle.jdbc.xa.client.OracleXADatasource</driver-name>
    <properties>
      <property>
        <name>user</name>
        <value>medrec</value>
      </property>
    </properties>
    <password->
      <encrypted>{AES}fy0q41+FkMM+ZhcliHQTX21fDGIyK0vdNwHi1B8P528=</password-encrypted>
    </jdbc-driver-params>
    <jdbc-connection-pool-params>
      <initial-capacity>5</initial-capacity>
      <max-capacity>10</max-capacity>
      <capacity-increment>1</capacity-increment>
    </jdbc-connection-pool-params>
    <jdbc-data-source-params>
      <jndi-name>jdbc/MedRecGlobalDataSourceXA</jndi-name>
      <global-transactions-protocol>TwoPhaseCommit</global-transactions-protocol>
    </jdbc-data-source-params>
  </jdbc-data-source>
```

How Data Source Connection Pools Are Used

A client retrieves a data source through a JNDI lookup and uses it to obtain a database connection.



ORACLE

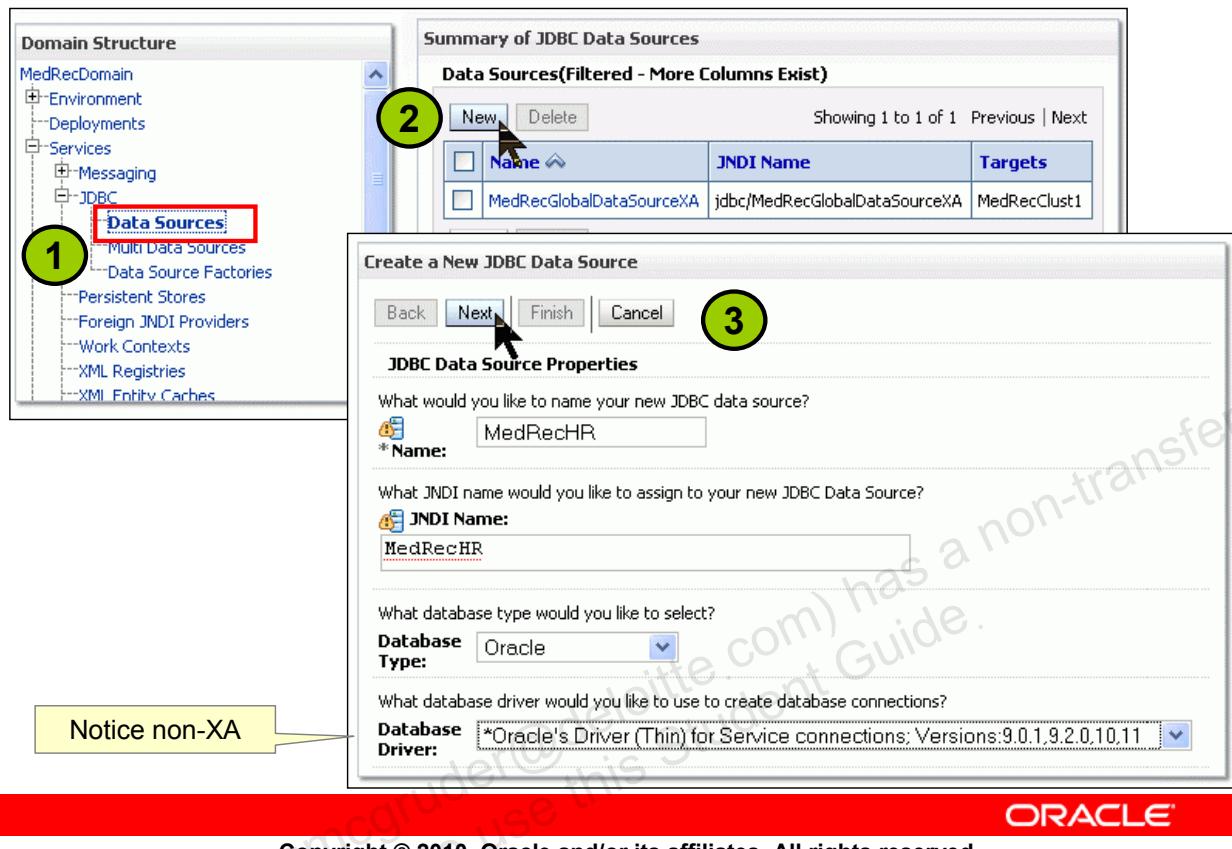
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

How Data Source Connection Pools Are Used

This is an example of how a data source is used by the client. The sequence of events is as follows:

1. A client performs a lookup in the Oracle WebLogic Server JNDI tree.
2. The client retrieves a data source object. A data source object contains a reference to the connection pool.
3. After a data source object is obtained, the client tries to get a database connection.
4. A specific connection is returned from the pool. The pool may have to be extended to create a new free connection, or there might not be any more free connections.
5. The connection then directly accesses the database through the connection.

Creating a Data Source Using the Administration Console



ORACLE

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

Creating a Data Source Using the Administration Console

You can create data sources via the Administration Console (as shown here) or WLST. Make sure that the JDBC drivers that you want to use to create database connections are installed on all the servers on which you want to configure database connectivity. Some JDBC drivers are installed with Oracle WebLogic Server, including the WebLogic Type 4 JDBC drivers for DB2, Informix, MS SQL Server, Oracle, and Sybase.

1. In the Domain Structure tree, expand Services > JDBC and then select Data Sources.
2. On the “Summary of Data Sources” page, click New.
3. In the JDBC Data Source Properties section of the “Create a New JDBC Data Source” page, enter or select the following information and click Next:
 - **Name:** Enter a configuration name for this JDBC data source.
 - **JNDI Name:** Enter the JNDI path to which this JDBC data source will be bound. Applications look up the data source on the JNDI tree by this name when reserving a connection.
 - **Database Type:** Select the database that you want to connect to. If your DBMS is not listed, select Other.
 - **Database Driver:** Select the JDBC driver that you want to use to connect to the database. The list includes common JDBC drivers for the selected DBMS. For example, the non-XA driver was selected, but you could have selected the XA driver. The non-XA will show an extra page for configuration.

Non-XA Configuration

This appears only if a non-XA driver was selected previously.

Create a New JDBC Data Source

Back **Next** Finish Cancel

Transaction Options

You have selected non-XA JDBC driver to create database connection in your new data source.

Does this data source support global transactions? If yes, please choose the transaction protocol for this data source.

Supports Global Transactions

Select this option if you want to enable non-XA JDBC connections from the data source to participate in global transactions using the *Logging Last Resource* (LLR) transaction optimization. Recommended in place of Emulate Two-Phase Commit.

Logging Last Resource

Select this option if you want to enable non-XA JDBC connections from the data source to emulate participation in global transactions using JTA. Select this option only if your application can tolerate heuristic conditions.

Emulate Two-Phase Commit

Select this option if you want to enable non-XA JDBC connections from the data source to participate in global transactions using the one-phase commit transaction processing. With this option, no other resources can participate in the global transaction.

One-Phase Commit

ORACLE

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

Non-XA Configuration

If you selected a non-XA JDBC driver, you are presented with two transaction options: Supports Global Transactions and Supports Local Transactions. If you select the non-XA option, WebLogic can use several alternative strategies to emulate XA on your non-XA driver.

Data Source Connection Properties

The screenshot shows the 'Create a New JDBC Data Source' dialog box. At the top, there are four buttons: Back, Next (highlighted with a mouse cursor), Finish, and Cancel. Below the buttons, the title 'Connection Properties' is displayed. A note says 'Define Connection Properties.' The first field is 'Database Name:' with the value 'orcl.us.oracle.com'. A yellow callout box points to this field with the text 'Not just "name." Be mindful of whether you are entering a service name vs. instance name vs. database name.' The next field is 'Host Name:' with the value 'localhost'. The third field is 'Port:' with the value '1521'. The fourth field is 'Database User Name:' with the value 'HR'. A yellow callout box points to this field with the text 'Sample schemas'. The fifth field is 'Password:' with two dots indicating the password. The sixth field is 'Confirm Password:' with two dots indicating the confirmation. At the bottom right of the dialog box is the 'ORACLE' logo.

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

Data Source Connection Properties

In the Connection Properties section of the “Create a New JDBC Data Source” page, enter values for the following properties and click Next:

- **Database Name:** This field is *overloaded*, which means there are multiple kinds of information that could go in this field depending on the context. It is not always the name of the database that you want to connect to. Exact database name requirements vary by JDBC driver and by DBMS. If you used Oracle’s Driver for Service Connections, the *service* name would be the full name `orcl.example.com`; if you used Oracle’s Driver for Instance Connections, the *instance* name would be just `orcl`. RAC naming is different as well. In any case for Oracle, it is not the *database* name.
- **Host Name:** Enter the DNS name or IP address of the server that hosts the database.
- **Port:** Enter the port on which the database server listens for connections requests. For Oracle databases, you can verify this by entering `lsnrctl status`.
- **Database User Name:** Enter the database user account name that you want to use for each connection in the data source.
- **Password/Confirm Password:** Enter the password for the database user account.

Test Configuration

The screenshot shows two panels of a wizard:

- Left Panel (Test Configuration):**
 - Header: "Create a New JDBC Data Source" with buttons: Back, Next, Finish, Cancel. The "Test Configuration" button is highlighted with a red box and a mouse cursor.
 - Test Database Connection:** Subtitle: "Test the database availability and the connection properties you provided." Description: "What is the full package name of JDBC driver class used to create database connections in the connection pool?" Driver Class Name: "oracle.jdbc.OracleDriver".
 - URL:** "jdbc:oracle:thin:@localhost:1521:HR".
 - Database User Name:** "HR".
 - Password:** "*****".
 - Confirm Password:** "*****".
 - Properties:** "user=HR".
 - Test Table Name:** "SQL SELECT 1 FROM DUAL".
- Right Panel (Select Targets):**
 - Header: "Create a New JDBC Data Source" with buttons: Back, Next, Finish, Cancel. The "Finish" button is highlighted with a red box and a mouse cursor.
 - Select Targets:**
 - Servers:** "MedRecAdmSvr" (unchecked), "MedRecSvr3" (checked).
 - Clusters:** "MedRecClust1" with three options: "All servers in the cluster" (radio button selected), "Part of the cluster" (radio button selected), "MedRecSvr2" (unchecked), "MedRecSvr1" (unchecked).

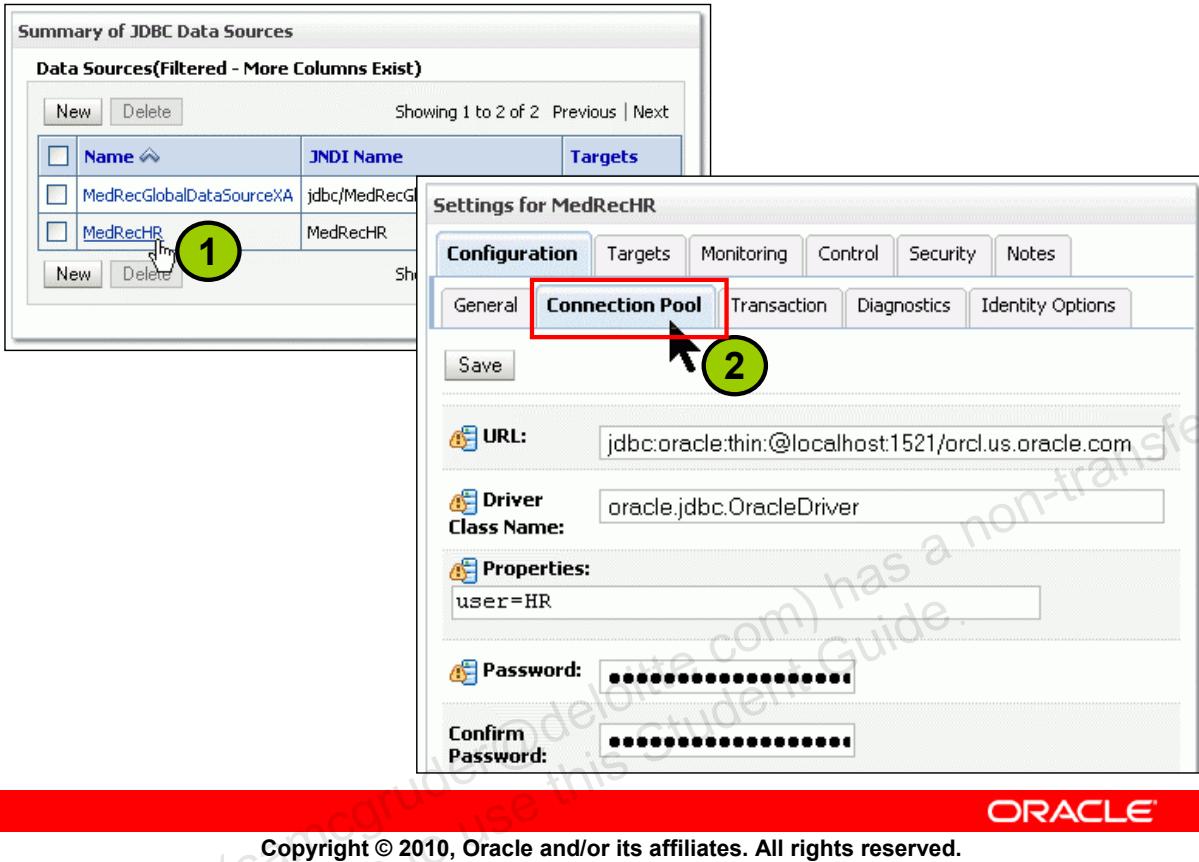
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Test Configuration

In the Test Database Connection section of the “Create a New JDBC Data Source” page, review the connection parameters and click Test Configuration. WebLogic attempts to create a connection from the Administration Server to the database. Results from the connection test are displayed at the top of the page. If the test is unsuccessful, you should correct any configuration errors and retry.

Selecting a target is optional. You can click Finish after testing without assigning a target. The JDBC source will be configured, but not deployed. If you skip selecting the target, there is a chance to deploy the JDBC source later. Select a server target (or not), and then click Finish.

Connection Pool Configuration



Connection Pool Configuration

The screenshot in the slide shows how you can modify a connection pool after the data source is created.

Before modifying a connection pool, you should know:

- The JDBC URL of the database
- The connection properties used to authenticate a user or optionally configure the driver
- The maximum number of connections that your application will be allowed by the DBA

After creating your initial data source configuration in the console, you can tune its connection pool settings:

1. In the Domain Structure tree, expand Services > JDBC and then select Data Sources. After selecting your data source, select Configuration > Connection Pool.
2. Enter values for the available data source attributes.

Note: The exclamation mark in a yellow triangle means that changing these values requires restarting some components.

Connection Pool Advanced

The screenshot shows the 'Connection Pool Advanced' configuration page. It includes fields for Initial Capacity (1), Maximum Capacity (15), Capacity Increment (1), Statement Cache Type (LRU), Statement Cache Size (10), and an 'Advanced' section containing Test Connections On Reserve (unchecked), Test Frequency (120), Test Table Name (SQL SELECT 1 FROM DUAL), Seconds to Trust an Idle Pool Connection (10), Shrink Frequency (900), Init SQL, Connection Creation Retry Frequency (0), and Login Delay (0). Callout boxes explain: 'Connection pool size' points to Maximum Capacity; 'Grows pool when more connections are needed' points to Capacity Increment; 'Periodically tests for bad connections and closes' points to Test Frequency and Test Table Name; 'Periodically closes idle connections' points to Shrink Frequency; and 'More options not shown' points to the bottom of the Advanced section.

Initial Capacity: 1

Maximum Capacity: 15

Capacity Increment: 1

Statement Cache Type: LRU

Statement Cache Size: 10

Advanced

Test Connections On Reserve

Test Frequency: 120

Test Table Name: SQL SELECT 1 FROM DUAL

Seconds to Trust an Idle Pool Connection: 10

Shrink Frequency: 900

Init SQL:

Connection Creation Retry Frequency: 0

Login Delay: 0

Connection pool size

Grows pool when more connections are needed

Periodically tests for bad connections and closes

Periodically closes idle connections

More options not shown

ORACLE

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

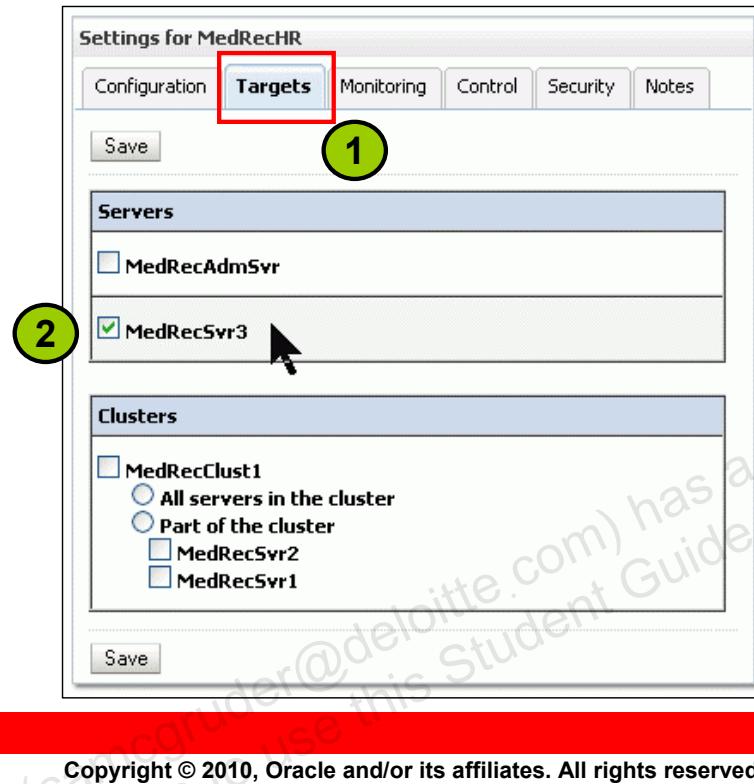
Connection Pool Advanced

Some of the key options are found under the Advanced section, including:

- **Initial Capacity:** This is the number of physical connections to create when deploying the connection pool. This is also the minimum number of physical connections that the connection pool will keep available.
- **Maximum Capacity:** This is the maximum number of physical connections that this connection pool can contain. For optimal performance, set the value of Initial Capacity equal to the value for Maximum Capacity, although that disables the dynamic resizing.
- **Capacity Increment:** When there are no more available physical connections to satisfy connection requests, Oracle WebLogic Server creates this number of additional physical connections and adds them to the connection pool up to the maximum capacity.
- **Test Frequency:** This is the number of seconds between when Oracle WebLogic Server tests unused connections. This requires that you specify a Test Table Name. DUAL is included in all Oracle database installations for such a purpose as this. Connections that fail the test are closed and reopened to reestablish a valid physical connection. If the test fails again, the connection is closed.

Targeting a Data Source

Deploy data sources to one or more servers in your domain.



ORACLE

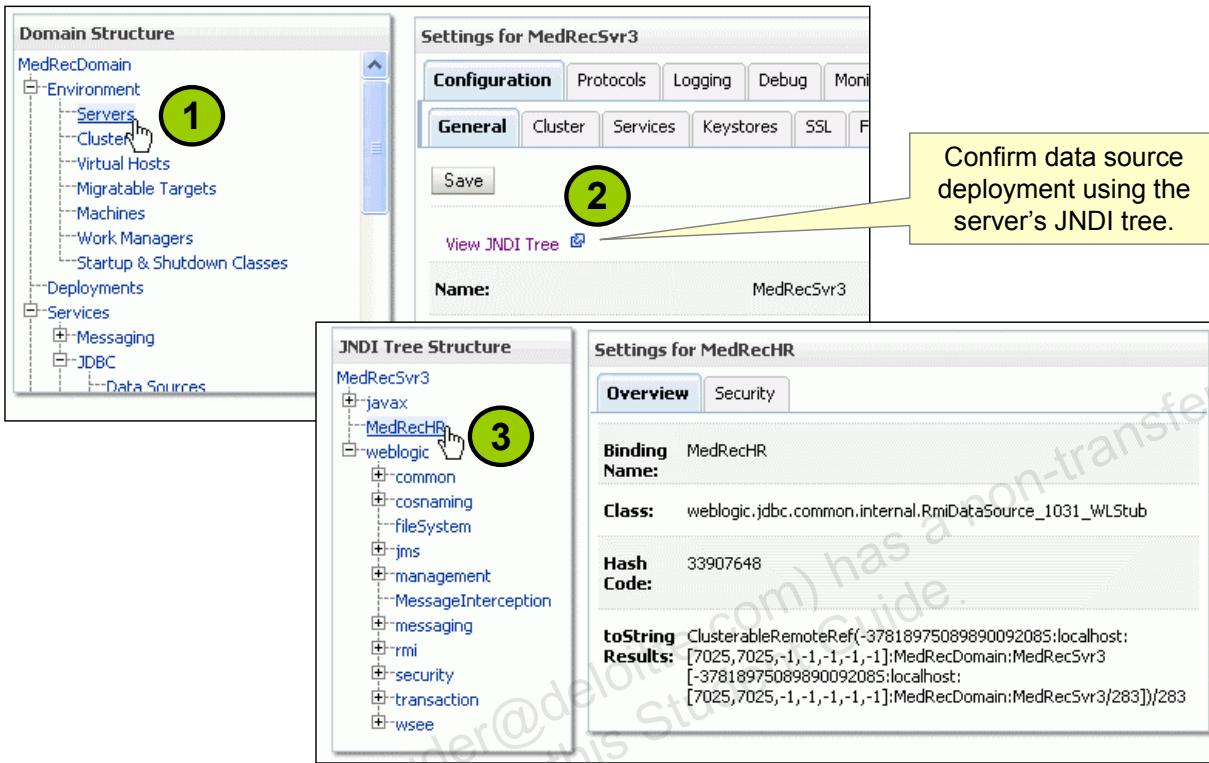
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Targeting a Data Source

This is the second opportunity to deploy a JDBC to a target. Any previous targets are prechecked when this page is displayed. When you target a JDBC data source, a new instance of the data source is created on the target. When you select a server as a target, an instance of the data source is created on the server. When you select a cluster as a target, an instance of the data source is created on all member servers in the cluster.

1. Navigate to the data source that you want to modify and click the Targets tab.
2. Select each server or cluster on which you want to deploy the data source, and click Save.

Viewing the Server JNDI Tree via the Administration Console



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

ORACLE

Viewing the Server JNDI Tree via the Administration Console

The screenshot in the slide shows viewing the Java Naming and Directory Interface (JNDI) tree. If the data source is deployed successfully, a new entry should be added to the local JNDI tree of the target servers. The name of the entry should match the JNDI name that is used to configure the data source. If you use a fully qualified JNDI name containing path separators (for example, “`hr.datasource.HRDataSource`” instead of just `MedRecHR`), the entry will not be found at the root of the tree. Instead, directories are created to match the fully qualified name, if they do not already exist, with plus and minus icons to expand and collapse them.

1. In the left pane, expand Environment > Servers. Then select a specific server.
2. On the default Configuration > General tab of the server, click View JNDI Tree. The JNDI tree is displayed in a new browser window or browser tab.
3. Use the left panel to navigate the directories of the JNDI tree.

Note: When you create contexts and bind objects programmatically, the subcontext will not be autocreated (therefore, subcontexts must be programmatically created before objects are placed into them); but when a JNDI entry is configured using the Administration Console as shown here, then the subcontext will be automatically created for you.

Listing the JNDI Contents Via WLST

- WLST provides a command-line utility for viewing the JNDI bindings.
- `jndi()` changes to the JNDI tree and `ls()` lists the bindings.

```
wls:/offline> connect("weblogic","welcome1","t3://localhost:7020")
wls:/base_domain/serverConfig> jndi()
wls:/base_domain/jndi> cd('AdminServer')

wls:/base_domain/jndi/AdminServer> ls()
dr-- ejb
dr-- javax
dr-- weblogic
-r-- cgDataSource
-r-- cgDataSource-nonXA
-r-- mejbmejb_jarMejb_EO
-r-- samplesDataSource
                                         weblogic.rmi.cluster.ClusterableRemoteObject
                                         weblogic.rmi.cluster.ClusterableRemoteObject
                                         weblogic.rmi.cluster.ClusterableRemoteObject
                                         weblogic.rmi.cluster.ClusterableRemoteObject
```

JDBC data source



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

Listing the JNDI Contents Via WLST

`jndi()` navigates to the JNDI tree for the server to which the Oracle WebLogic Scripting Tool (WLST) is currently connected. This read-only tree holds all the elements that are currently bound in JNDI.

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

The following example navigates from the run-time MBean hierarchy to the domain JNDI tree in an administration server instance.

```
wls:/myserver/runtime> jndi()
Location changed to jndi tree. This is a read-only tree with No
root. For more help, use help('jndi')
wls:/myserver/jndi> ls()
dr-- ejb
dr-- javax
dr-- jms
dr-- weblogic
```

Demonstration

- Configure data sources for Oracle Database.
- Go to OTN > Tutorials > Fusion Middleware > Oracle WebLogic Server 10.3 > Deployment of Applications > [Configure Data Sources.](#)

ORACLE

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

Demonstration

See the demonstration at the following URL:

http://www.oracle.com/technologyobe/fusion_middleware/wls103/appdeploy/configure/datasource/Conf_DS_WLS.htm

JDBC URLs

Database locations are specified using a JDBC Uniform Resource Locator (URL).

- Example 1:
 - This URL specifies that the `oracle:thin` subprotocol should be used to connect to an Oracle database:

```
jdbc:oracle:thin:@dbhost:1521:SALESINFO
```

- Example 2:
 - This URL can be used to access a PointBase database:

```
jdbc:pointbase:server://dbhost:9092/HRDATABASE
```

ORACLE

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

JDBC URLs

If you use a JDBC driver developed by a third party, the documentation tells you what subprotocol to use—that is, what to put after “`jdbc:`” in the JDBC URL. The syntax for a JDBC URL is `jdbc : subprotocol : subname`.

- `subprotocol` identifies the database connectivity mechanism.
- `subname` identifies the data source. The subname can vary depending on the subprotocol.

The contents and syntax of subname depend on subprotocol. subname can also specify a network address for the database—for example, subname can be specified using “`//hostname:port/dbname`.”

For Example 1

- `dbhost` : The host name or IP address
- `1521` : The default listener port
- `SALESINFO` : The system identifier (SID), the name of the database

For Example 2

- `subprotocol` is `pointbase:server`.
- `subname` is a location of the PointBase database named `HRDATABASE`.

Connection Properties

- Are key/value pairs
- Are used to configure JDBC connections
- Are passed to the driver during connection setup

ORACLE

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

Connection Properties

Connection properties are a set of key/value pairs that are passed to the driver when database connections are created. Connection properties are specific to the driver. For a complete list, see your driver documentation.

Specifying Connection Properties

A partial list of connection properties for the supplied drivers:

Driver	Some Connection Properties
Oracle	User, Password, ServerName, ServiceName, PortNumber
Sybase	User, Password, ServerName, DatabaseName, PortNumber
MSSQL	User, Password, ServerName, DatabaseName, PortNumber
Informix	User, Password, ServerName, DatabaseName, PortNumber
PointBase	cache.size, crypto.communication, database.home, database.pagesize



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

Specifying Connection Properties

PointBase connection properties can be set in the pointbase.ini file. You can select the pointbase.ini parameters to configure the database properties. By configuring the database properties, you can increase the performance of your system. However, PointBase should not be used in a production environment, so performance for that DBMS is less critical.

Road Map

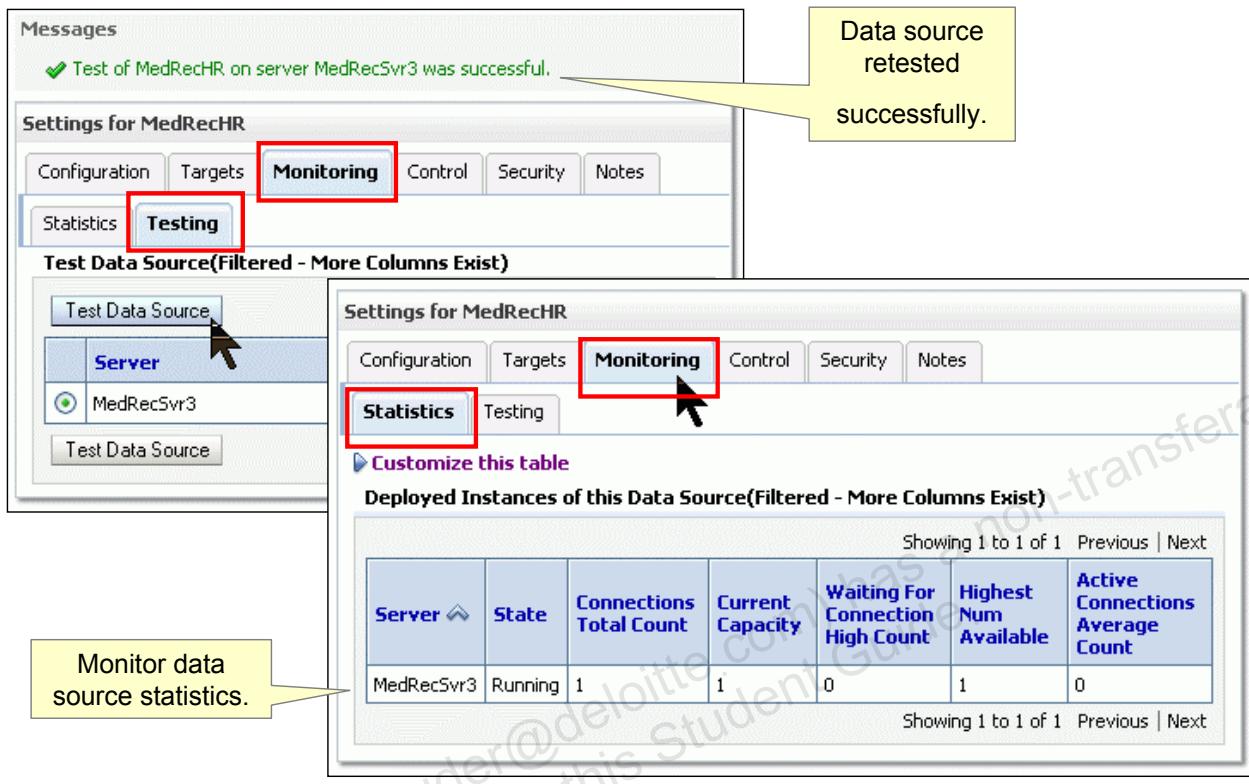
- Overview of JDBC
- Data sources
- Monitoring and testing data sources
 - Monitoring
 - Testing
 - Suspend/resume



ORACLE

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

Monitoring and Testing a Data Source



ORACLE

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

Monitoring and Testing a Data Source

After you create a JDBC data source and target it to one or more servers, you can monitor it in the Administration Console. Locate and select your new data source and select Monitoring > Statistics. Statistics are displayed for each deployed instance of the data source. Optionally, click “Customize this table” to change the columns displayed in the Statistics table. For example, some of the available columns (not displayed by default) include:

- **Active Connections Current Count:** The number of connections currently in use by applications
- **Active Connections Average Count:** The average number of active connections from the time that the data source was deployed
- **Connections Total Count:** The cumulative total number of database connections created in this data source from the time that the data source was deployed
- **Current Capacity:** The current count of JDBC connections in the connection pool in the data source
- **Highest Num Available:** The highest number of database connections that were available at any time in this instance of the data source from the time that the data source was deployed
- **Waiting for Connection High Count:** The highest number of application requests concurrently waiting for a connection from this instance of the data source

Connection Pool Life Cycle

For this data source...

...on a given server...

...take this action.

Settings for testSample

Control

Customize this table

Deployed Instances of this Data Source

Server Name	State	Status of Last Action
MedRecSvr1	Running	None

Suspend | Resume | Shutdown | Start

Suspend | Force Suspend

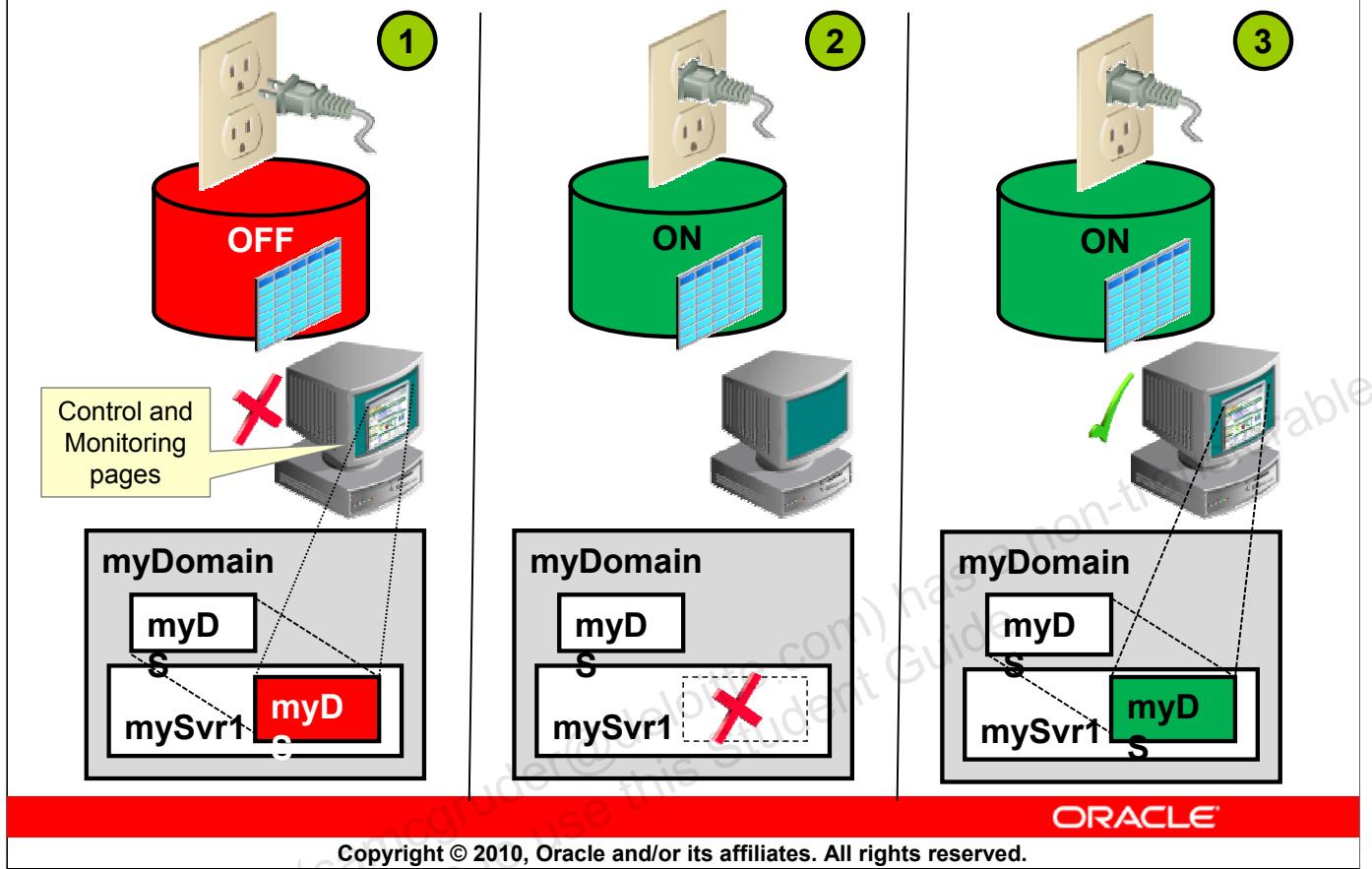
ORACLE

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

Connection Pool Life Cycle

By default, a connection pool is automatically started when it is deployed. You can manually stop and restart the connection pool. This might be necessary if you change the username/password or some other characteristic of the connection. If you wanted to gracefully shut down an application, you might start by shutting down the connection pool.

Fixing an Offline Data Source



Fixing an Offline Data Source

If you start your domain managed servers while the Database is *not* available, the Data Source instance does *not* show up on the Control and Monitoring pages. So you cannot select it for Resume nor Start operations and it will not be found in the JNDI tree.

You can start the Database and then do either of the following options:

1. Restart the managed servers (which could take a long time and you may have application downtime) or
2. A better way is to wait for the database to come back up, then Lock & Edit > Remove the Data Source from Target > Save > Activate.
This is a complete undeployment of your Data Source on a running server. The Data Source remains defined to the domain though, just not targeted.
3. Then you redeploy it again with: Lock & Edit > Put Data Source again on its Target(s) > Save > Activate.
This is a clean restart of your Data Source on a running WLS.

If this is a typical situation for some company (starting WLS when Database is unavailable), you also can set Initial Capacity = 0, that means the Data Source Java object is created and bound to the JNDI tree, but no connections are created. The instance shows up in the JNDI tree *and* on the Control and Monitoring pages. You have to set Capacity Increment ≥ 1 and then connections are created on the first client request.

Quiz

Which is NOT an available configuration attribute for a JDBC data source?

- a. Host name
- b. Queue size
- c. Test frequency
- d. Initial capacity
- e. Capacity increment



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

Quiz

Which are the two levels of data sources available in Oracle WebLogic Server?

- a. Connection
- b. Web
- c. Application
- d. Process
- e. System



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

Quiz

Client applications look up data sources from the local server's
_____ tree:

- a. Application
- b. Web
- c. LDAP directory
- d. JNDI
- e. System

ORACLE

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

Summary

In this lesson, you should have learned how to:

- Define JDBC high-level architecture
- Configure Oracle WebLogic Server–provided JDBC driver types
- Create data source definitions
- Create connection pool definitions
- Manage JDBC resources using the Administration Console



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

Practice 14 Overview: Configuring JDBC Data Sources

This practice covers the following topics:

- Creating JDBC modules (via GUI and WLST)
- Deploying JDBC modules
- Testing JDBC modules



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

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

Carl McGruder (camcgruder@deloitte.com) has a non-transferable
license to use this Student Guide.

15

Setting Up Java Message Service (JMS) Resources

ORACLE

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

Objectives

After completing this lesson, you should be able to:

- Describe JMS
- Describe how Oracle WebLogic Server JMS is implemented
- Configure JMS server
- Configure connection factories
- Configure queues and topics
- Configure persistent messages
- Deploy an application that uses JMS
- Monitor JMS resources and messages



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

Objectives

Scenario

Consider an online order entry application that integrates with a shipping application. In this case, you may not want the online customer to keep waiting for the shipping application to finalize the shipping process.

Generally, in such cases, the following steps are performed:

1. The customer places an order using the order entry application.
2. When the order is completed and confirmed (may involve a credit check and so on), the order details are placed in a message queue.
3. The shipping application regularly checks the order message queue, picks up the orders from the message queue, assigns the appropriate shipping agency (for example, UPS, FedEx, or USPS), and appropriately generates shipping labels.
4. In addition, the shipping may append the shipping details to the order message.

Road Map

- Oracle WebLogic Server JMS administration
 - JMS overview
 - JMS server and modules
 - Types of JMS destinations
- Configuring JMS objects
- Durable subscribers and persistent messaging
- Monitoring JMS



ORACLE

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

Message-Oriented Middleware

- The message-oriented architecture enables asynchronous and cross-platform integration of applications.
- Message-oriented middleware refers to an infrastructure that supports messaging.
- Typical message-oriented middleware architectures define the following elements:
 - Message structure
 - The way to send and receive messages
 - Scaling guidelines



ORACLE

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

Message-Oriented Middleware

The message-oriented middleware became widely used when providers created architectures that could operate in a standard way on a variety of platforms and enabled asynchronous communication between applications. These providers gained popularity in enabling integration of mainframes and personal computers.

Even though there is much competition and variety in message-oriented middleware products, they tend to fall into one of the following categories:

- Point-to-point
- Publish/Subscribe
- Request-reply

JMS Messaging Models

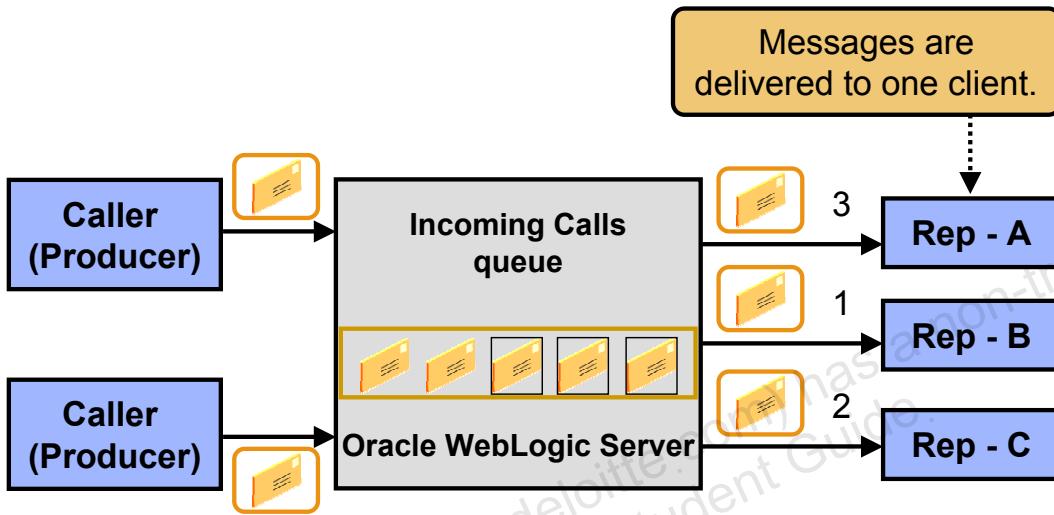
JMS supports the point-to-point (PTP) and Publish/Subscribe messaging models. The models are very similar, except the following:

- The PTP messaging model enables delivery of a message to exactly one recipient.
- The Publish/Subscribe messaging model enables delivery of a message to multiple recipients.

The request-reply messaging model is more suited in a synchronous messaging environment where the requester and replier are in conversational mode—the requester waits for a response from the replier before continuing work. It is not explicitly supported in JMS.

Point-To-Point Queue

Many message producers can serialize messages to multiple receivers in a queue.



ORACLE

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

Point-To-Point Queue

When using a PTP queue, multiple message producers can put messages onto a single queue. The queue serializes the messages in a linear order. Multiple receivers can take messages off the queue; the messages typically come off in a first-in, first-out (FIFO) order; the oldest message on the queue is the first one to be taken off.

A message can be delivered only to one receiver. Receivers are also referred to as consumers.

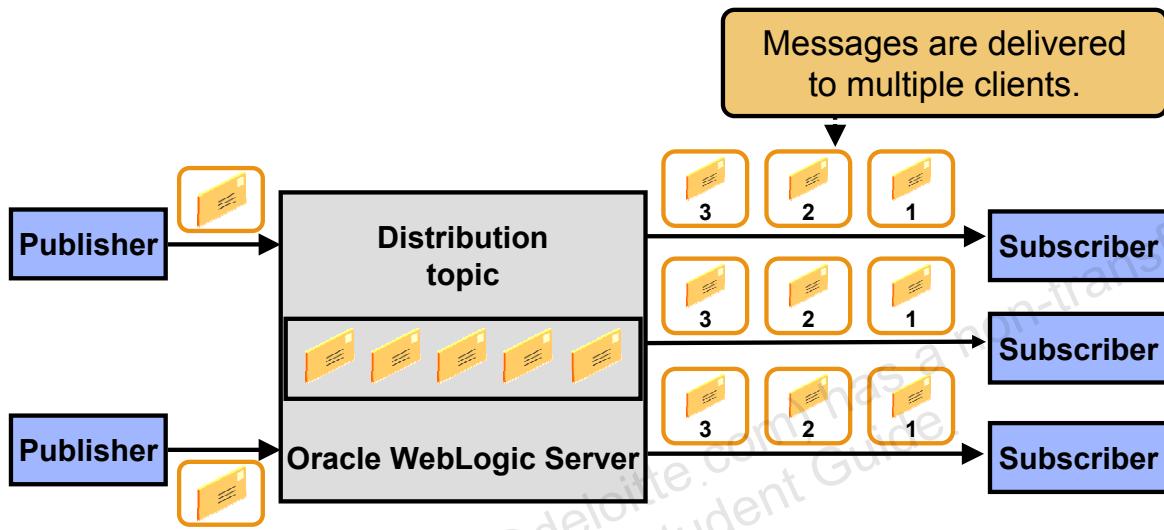
An example of when to use a PTP queue would be at a call center.

- Calls are routed into the network through a PBX. The PBX system places incoming calls onto an Incoming Calls queue. When a service representative is available, the representative requests for the next caller in the system.
- The system pulls off the queue the caller who has been waiting the longest (FIFO method) and routes the caller to the service representative.
- After the conversation is established between an in-queue customer and a representative, it becomes a synchronous communication. (This is similar to request-reply mode).

This is only an example and, in many cases, the responses are not just pure FIFO but weightings assigned by the organizations.

Publish/Subscribe Topics

Publishing and subscribing to a topic decouples producers from consumers.



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

ORACLE

Publish/Subscribe Topics

Having the publishers publish to a topic rather than directly to a list of subscribers decouples the publishers and subscribers.

By doing this, a publisher is not required to manage the number of subscribers (if any) that must receive the message. By delegating the message delivery work to the message-oriented middleware server (which manages the topic), the publisher does not have to manage the delivery of guaranteed messages, fault tolerance of its production, load balancing, or other issues. By decoupling a subscriber from the publisher, the subscriber does not have to determine whether its publisher is active. If the message-oriented middleware server is executing, the needs of both the publishers and the subscribers are met.

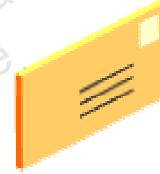
An example of using a Publish/Subscribe topic is a stock ticker application.

- A typical system would set up a topic for each stock that is traded on the exchanges.
- When a trade is made on a stock, the respective exchange publishes a message to the topic that is associated with the stock traded.
- Clients who are interested in receiving updates about the status of their stocks use a program to subscribe to the topics of each stock they are interested in.
- When the topic update is recognized, the message server broadcasts the message to all the interested (clients) stock ticker programs.

Oracle WebLogic Server JMS Features

Oracle WebLogic Server JMS supports:

- Both the point-to-point and Publish/Subscribe JMS models
- Acknowledgement-based guaranteed delivery
- Transactional message delivery
- Durable subscribers
- Distributed destinations
- Recovery from failed servers



ORACLE

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

Oracle WebLogic Server JMS Features

An enterprise messaging system enables applications to asynchronously communicate with one another through the exchange of messages. A message is a request, report, and/or event that contains information needed to coordinate communication between different applications. A message provides a level of abstraction, which allows you to separate the details about the destination system from the application code.

The Oracle WebLogic Server implementation of JMS fully supports the point-to-point and Publish/Subscribe models of the messaging middleware.

Oracle WebLogic Server also provides acknowledgement-based (ACK) guaranteed message delivery (GMD) by enabling persistent storage of messages until the receiver of the message issues an acknowledgement of receipt.

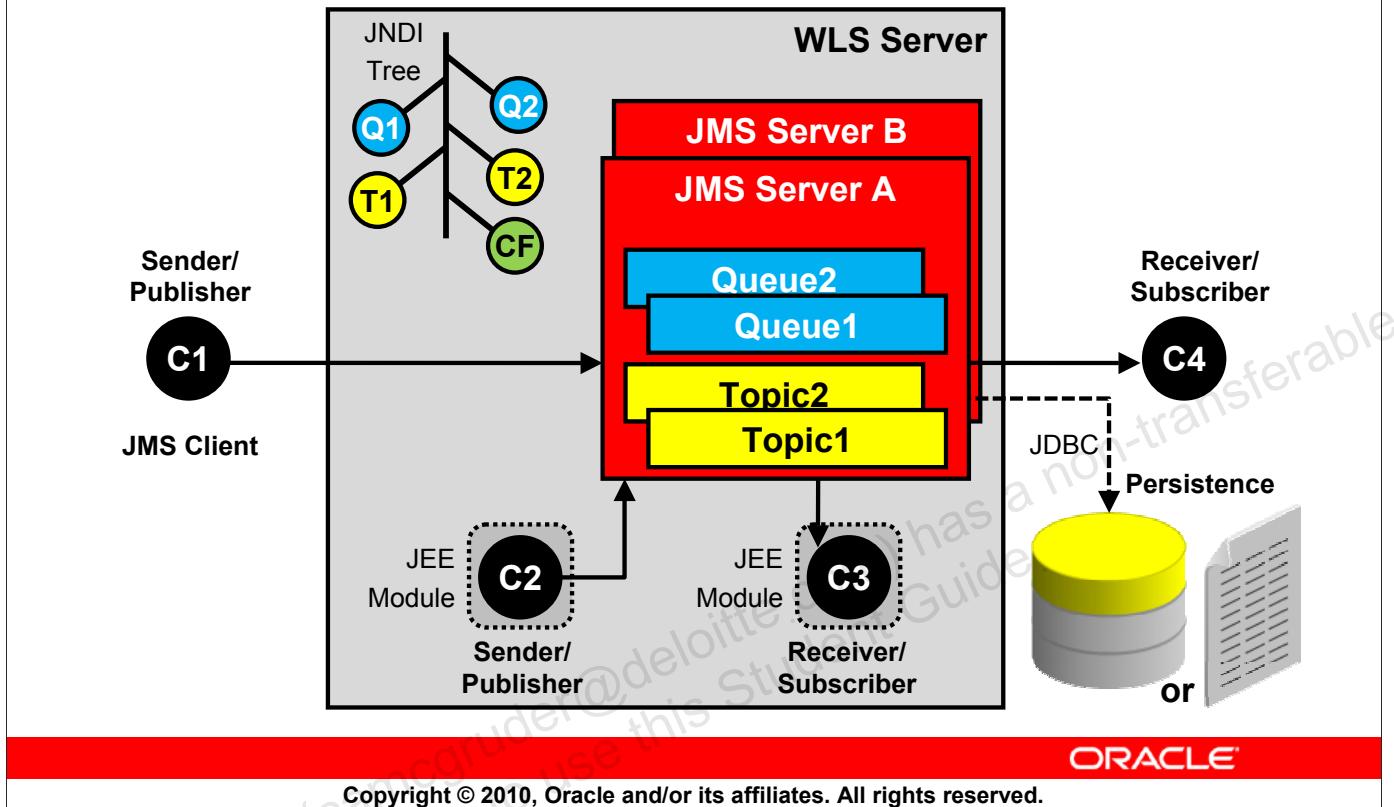
Oracle WebLogic Server JMS uses its built-in support for JDBC and JDBC connection pools to persist JMS messages in a database.

Oracle WebLogic Server supports transactional message delivery. Transactional message delivery gives the developer the ability to put a JMS session into a transaction context. In Oracle WebLogic Server JMS, the message is not visible or available for consumption until the transaction is committed. A session can optionally roll back the transaction, which has the transaction “drop” the messages it had previously buffered.

Oracle WebLogic Server JMS Features (continued)

Oracle WebLogic Server allows clients to register themselves as durable subscribers. A durable subscriber is a client that expects to receive all persistent messages that are sent to a particular destination, whether the client is currently executing or not. If the durable subscriber is not currently executing, Oracle WebLogic Server stores the messages in a persistent store until the durable subscriber reactivates and retrieves the stored messages.

WebLogic JMS Architecture



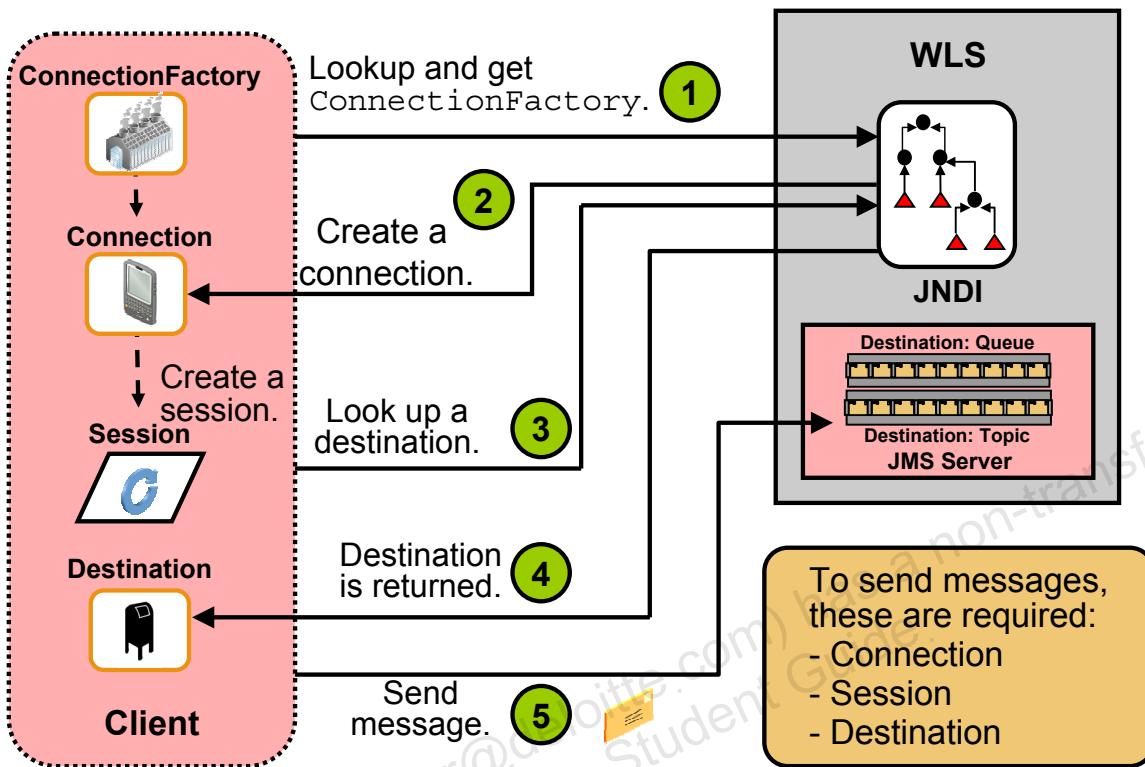
Oracle WLS JMS Architecture

The major components of the WebLogic JMS Server architecture include:

- JMS servers that can host a defined set of modules and any associated persistent storage that resides on a WebLogic Server instance. JMS server configuration is stored in the domain config.xml file.
- JMS modules that contain configuration resources (such as queues, topics, and connection factories) and are defined by XML documents that conform to the weblogic-jms.xsd schema
- Client JMS applications that either produce messages to destinations or consume messages from destinations
- Java Naming and Directory Interface (JNDI), which provides a resource lookup facility. JMS resources such as connection factories and destinations are configured with a JNDI name. The run-time implementations of these resources are then bound to JNDI using the given names.
- WebLogic persistent storage (file store or JDBC) for storing persistent message data

In the slide, C1 and C4 are remote JMS clients, C2 and C3 are server-side JMS clients. As a sample flow, C1 can do a JNDI lookup and download Connection Factory CF and Topic T1. Then C1 creates the Connection, Session, and Publisher object and publishes a message. C4 can do the same to receive the message.

Typical JMS Messaging Process



ORACLE

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

Typical JMS Messaging Process

In JMS implementations, developers use a connection factory to enable their applications to connect to a queue or topic.

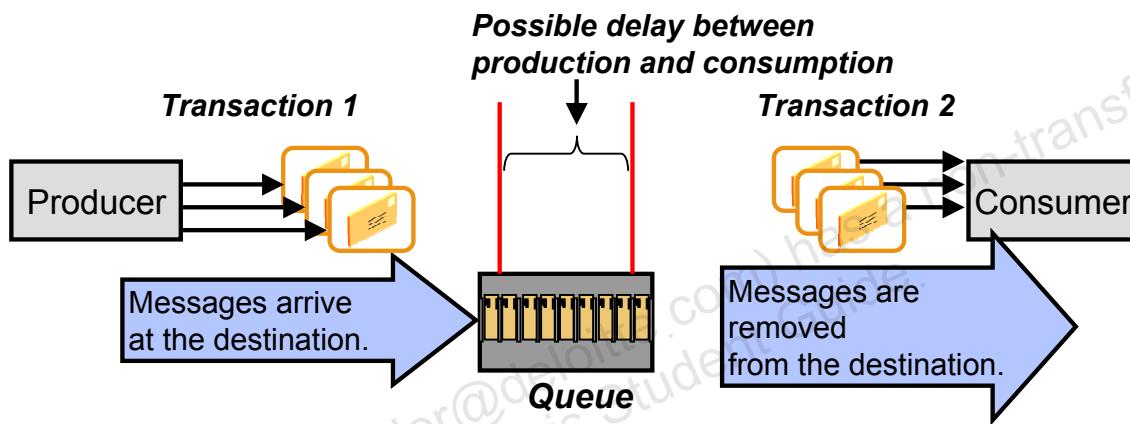
A connection factory is a lightweight object stored on a JNDI tree that is used to create connections to destinations. A connection is a communication link to the JMS server that is used to create sessions.

Sessions are bound to destinations to create senders, receivers, publishers, subscribers, and empty message objects. A session is also used to demarcate transactions.

Destination, a lightweight object stored on JNDI, is the target for the messages.

Transacted Messaging

- A JMS client can use Java Transaction API (JTA) to participate in a distributed transaction.
- Alternatively, a JMS client can demarcate transactions that are local to the JMS session through a transacted session.
- Participation in a transaction is optional.



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

Transacted Messaging

JMS clients can participate in a distributed or local transaction. There are two scenarios:

- On the Producer side, 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.
- On the Consumer side, a transaction begins and some operations, such as processing messages, are performed. If the transaction commits, the processed messages are removed from the destination. If the transaction rolls back, the messages stay in the destination.

Notice that the left red line of the Queue is indented by one message (yellow box) indicating that the producing transaction ends when the message reaches the queue. The right red line of the Queue is also indented by one message as per above. The consuming transaction begins when the consumer starts to remove the message from the queue.

The point is that while the message is on the queue, and not being produced or consumed, there is no transaction.

JMS Administrative Tasks

- Creating and monitoring JMS servers
- Creating connection factories
- Creating and monitoring destinations
- Creating JMS stores
- Configuring paging thresholds and quotas
- Configuring durable subscriptions
- Managing JMS service failover

ORACLE

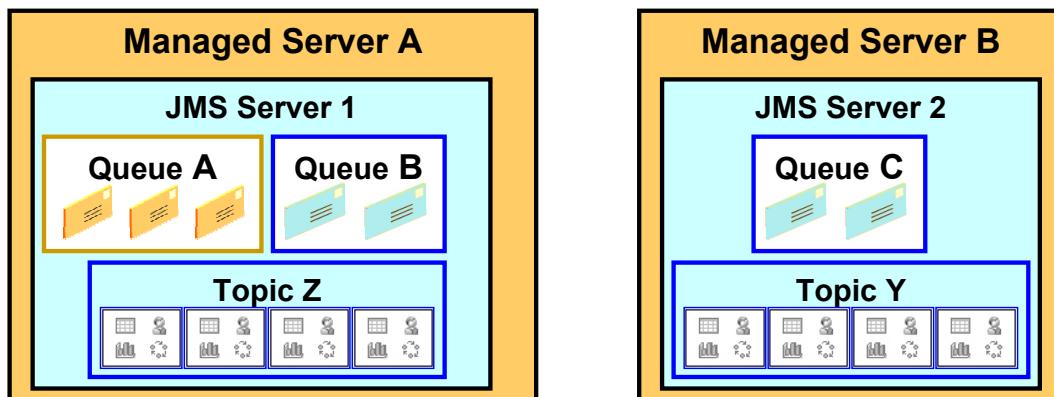
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

JMS Administrative Tasks

As an administrator, you are responsible for configuring and monitoring most aspects of JMS.

The architecture of your system determines the type of JMS destinations to configure. It is your responsibility to monitor the Oracle WebLogic Server JMS and gather statistics. All these administrative tasks are discussed throughout this lesson.

Oracle WLS JMS Implementation



Resource definitions: In JMS modules

JMS Module A

SubDeployment 1:

Queue A: Target (JMS Server 1)

SubDeployment 2:

Queue B: Target (JMS Server 1)

SubDeployment 3:

Topic Z: Target (JMS Server 1)

ORACLE

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

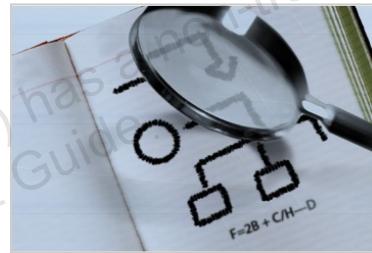
Oracle WLS JMS Implementation

When you implement JMS in WLS, you configure the following JMS resources:

- Configure the necessary JMS servers and target them to the appropriate managed servers.
- Configure JMS modules.
- Within the JMS modules, you define the queue or topic resources.
- Then using the subdeployment definitions, target the queues to the appropriate JMS servers. Queues and topics can be targeted to only a single JMS server, while Connection factories, uniform distributed queues, and uniform distributed topics can be targeted to multiple JMS servers or a cluster. Clusters are discussed in the lesson titled “Introduction to Clustering.”

Road Map

- Oracle WebLogic Server JMS administration
- Configuring JMS objects
 - Configuring JMS servers
 - Configuring JMS modules and subdeployments
 - Configuring connection factories
 - Configuring destinations
- Durable subscribers and persistent messaging
- Monitoring JMS



ORACLE

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

Creating a JMS Server

The screenshot shows the Oracle WebLogic Administration Console interface. On the left, the 'Domain Structure' tree is expanded to show 'MedRecDomain' with 'Environment', 'Deployments', 'Services' (selected), 'Messaging' (selected), and 'JDBC'. Under 'Messaging', 'JMS Servers' is highlighted with a green circle labeled '1'. In the center, the 'Summary of JMS Servers' page displays a table of existing JMS servers: 'MedRecJMS' and 'MedRecJMS'. A 'New' button is highlighted with a green circle labeled '2'. On the right, a 'Create a New JMS Server' dialog box is open. It has 'Next' and 'Cancel' buttons at the top. The 'JMS Server Properties' section contains a note about identifying the new server. The 'Name' field is populated with 'HRJMSServer' and is highlighted with a green circle labeled '3'. Below it, a 'Persistent Store' dropdown is set to '(none)'.

Creating a JMS Server

You can create and configure a JMS server by using the Administration Console. To create a JMS server, perform the following steps:

1. Expand the Services node in Domain Structure in the left panel, and then expand the Messaging node. Click JMS Servers. The summary of JMS servers appears in the right pane.
2. Click Lock & Edit to enable editing configuration. Then click New at the JMS Servers table. The “Create a New JMS Server” dialog box appears.
3. Enter values for the following configuration parameters:
 - Name: The name of the JMS server
 - Persistent Store: The backing store used by destinations. A value of none means that the JMS server will use the default persistent store that is configured on each targeted WLS instance.
4. Click Next to target a JMS server.
5. When you specify that you want to create a new store in step 3, the “Select store type” page appears. You can select File Store or JDBC Store.
 - If you specify File Store, the “File store properties” page appears. When creating a file store for the JMS Persistent store, the path name to the directory must exist on your system, so be sure to create it before completing this page.
 - If you selected JDBC Store, in the “Create new JDBC Store” page, select a configured JDBC data source or configure a new JDBC data source for the store. You cannot configure a JDBC data source that is configured to support global transactions.

Configuring a JMS Server

The screenshot shows the 'Summary of JMS Servers' page. A table lists JMS servers: HRJMServer, MedRecJMSServer_auto_1, and MedRecJMSServer_auto_2. The 'HRJMServer' row is selected and highlighted with a green circle labeled '1'. On the right, the 'Settings for HRJMServer' dialog is open. It has tabs for Configuration, Logging, Targets, Monitoring, Control, and Notes, with 'Configuration' selected. Sub-tabs include General, Thresholds and Quotas, and Session Pools, with 'General' selected. A 'Save' button is highlighted with a green circle labeled '2'. The 'Name:' field contains 'HRJMServer'. The 'Persistent Store:' dropdown is set to '(none)'. Under 'Paging Directory:', there is a checkbox for 'Paging File Locking Enabled' which is checked. The 'Message Buffer Size:' field contains '-1'.

ORACLE

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

Configuring a JMS Server

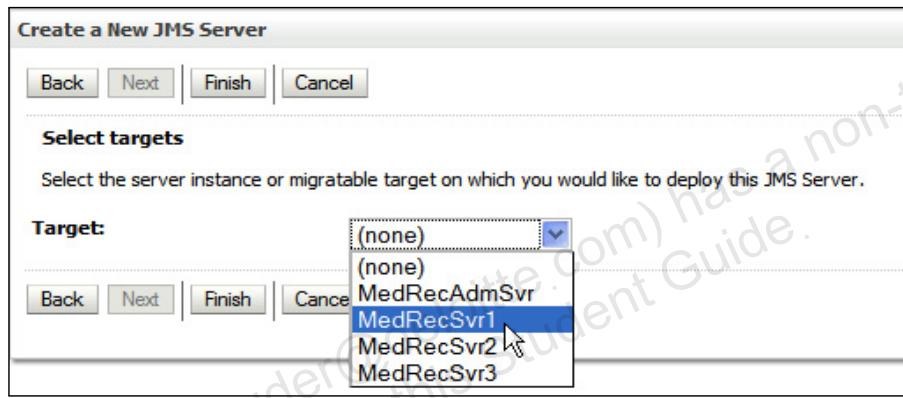
You can change the configuration of already created JMS servers or add configurations on a JMS server by performing the following steps:

1. Select Services > Messaging > JMS Servers from the Domain Structure pane. Locate and click the link to the JMS server that you want to configure.
2. Enter the values appropriately on the “Settings for HRJMServer” page.

You can set persistent stores at the time of creating a JMS server. If you have already configured persistent stores, you can assign one of them when configuring the JMS server.

Targeting a JMS Server to a Managed Server

- By appropriately selecting managed servers, you can target where the JMS queue or topic will be managed.
- The JMS server is associated with only one WebLogic Server instance.
- If you want a JMS server on each server in a cluster, you must configure a JMS server for each server.



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

ORACLE

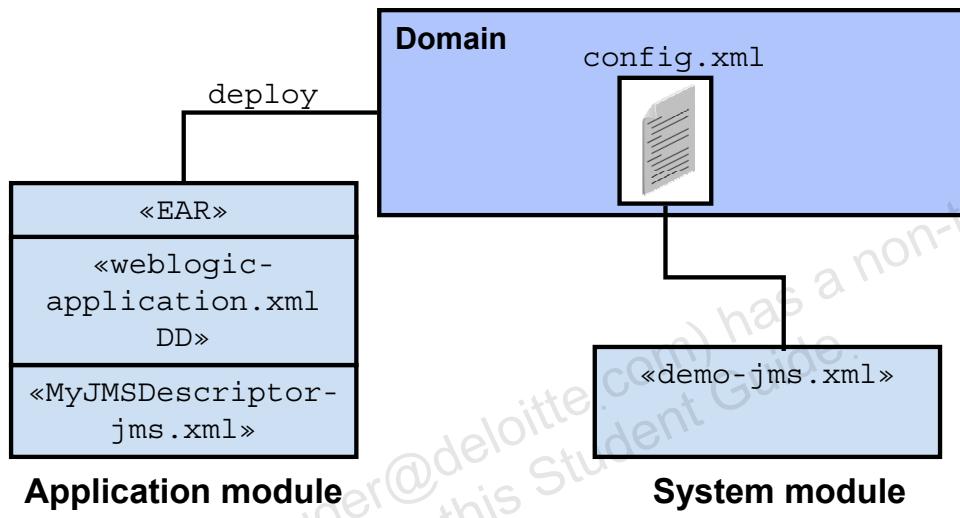
Targeting a JMS Server to a Managed Server

When the JMS server that you created is selected in the left pane, a dialog box appears in the right pane showing the tabs that are associated with this instance.

1. Select a target from the Target drop-down list.
(Note: A JMS server can be targeted to only one WebLogic Server.)
2. Click Finish.

JMS Modules

- JMS resources can be configured either as system modules or as application modules.
- As an administrator, you normally configure system modules.



ORACLE

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

JMS Modules

JMS modules are application-related definitions that are independent of the domain environment. You create and manage JMS resources either as system modules or application modules.

- JMS system modules are typically configured using the Administration Console or the WebLogic Scripting Tool (WLST), which adds a reference to the module in the domain's `config.xml` file. System modules are owned and modified by the WebLogic administrator and are available to all applications.
- JMS application modules are a WebLogic-specific extension of Java EE modules and can be deployed either with a Java EE application (as a packaged resource) or as stand-alone modules that can be made globally available. Application modules are owned and modified by WebLogic developers, who package JMS resource modules with the application's EAR file.

After the initial deployment is completed, an administrator has only limited control over the deployed applications. For example, administrators are allowed only to ensure the proper life cycle of these applications (deploy, undeploy, redeploy, remove, and so on) and tune parameters, such as increasing or decreasing the number of instances of any given application to satisfy the client needs. Other than life cycle and tuning, any modification to these applications must be completed by the application development team.

JMS Modules

- Configuration of JMS resources such as queues, topics, and connection factories are within JMS modules.
- Similar to other Java EE modules such as data sources, the configurations are in XML files that conform to the `weblogic-jms.xsd` schema.
- An administrator can create and manage JMS modules as:
 - Global system resources
 - Global stand-alone modules
 - Modules packaged with an enterprise application



ORACLE

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

JMS Modules (continued)

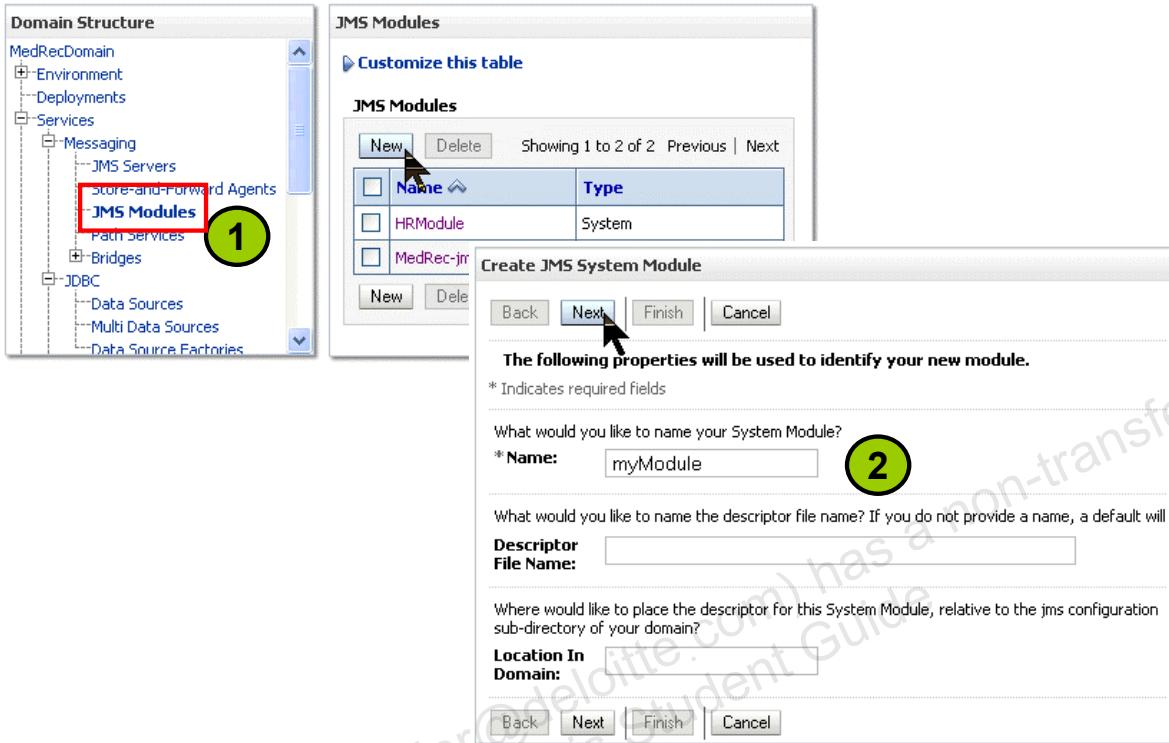
During the process of deploying a JMS application, you link the application components to the environment-specific JMS resource definitions, such as the server instances (deployment target) that should host a given application component, and the location to use for persisting JMS messages.

With modular deployment of JMS resources, you can migrate your application and the required JMS configuration from environment to environment, such as from a testing environment to a production environment, without opening an enterprise application file (such as an EAR file) or a stand-alone JMS module, and without extensive manual JMS reconfiguration.

JMS configuration resources, such as destinations and connection factories, are stored outside of the WebLogic domain configuration file as module descriptor files, which conform to the `weblogic-jms.xsd` schema. JMS modules do not include the JMS server definitions.

The JMS system modules must be targeted to one or more Oracle WebLogic Server instances or to a cluster. The targetable resources that are defined in a system module must also be targeted to a JMS server or the Oracle WebLogic Server instances within the scope of a parent module's targets. Additionally, the targetable JMS resources within a system module can be further grouped into *subdeployments* during the configuration or targeting process to provide further loose-coupling of the JMS resources in a WebLogic domain.

Creating a JMS Module



ORACLE

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

Creating a JMS Module

A JMS Module may be created by the administrator using the Administration Console, or by the programmer with an IDE and packaged with the application. To create a JMS System Module using the Administration Console, perform the following steps:

1. Select MedRecDomain > Services > Messaging > JMS Modules > New.
2. Enter a Name, and optionally a descriptor and location for the descriptor.
3. (Not shown) Select one or more Targets for this module, such as the Administration Server or one or more of the managed servers, or a cluster.
4. You can add modules (such as Connection factories, Queues, and Topics) at this time, or not. If you choose not to add modules now, you can come back at a later time to add or modify modules.

Modular JMS Resource Configuration and Deployment

- Modular deployment simplifies the task of migrating JMS resources between environments, such as from:
 - Development to integration
 - System test to production
- You can migrate your application and the required JMS configuration without:
 - Opening an EAR file
 - Extensive manual JMS reconfiguration

ORACLE

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

Modular JMS Resource Configuration and Deployment

A subdeployment for JMS destinations is a mechanism by which queues and topics, and possibly connection factories, are grouped and targeted to a single JMS server. Queues and topics depend on the JMS servers they are targeted to for the management of persistent messages, durable subscribers, and message paging. To reconfigure a subdeployment's targets, use the parent system module's subdeployment management page.

For example, if you want to co-locate a group of queues with a connection factory that is targeted to a specific JMS server, you can associate the queues with the subdeployment that the connection factory belongs to, provided that the connection factory is not already targeted to multiple JMS servers (for example, targeted to a server instance hosting multiple JMS servers).

Creating Packaged JMS Modules: You create packaged JMS modules using an enterprise-level integrated development environment (IDE) or a development tool that supports the editing of XML descriptor files. You then deploy and manage stand-alone modules using the JSR 88-based tools, such as the `weblogic.Deployer` utility or the WebLogic Administration Console.

Deploying a Packaged JMS Module: The deployment of packaged JMS modules follows the same model as all the other components of an application: individual modules can be deployed to a single server, a cluster, or individual members of a cluster.

Connection Factories

- JMS connection factories are used to set default client connection parameters, including:
 - Message priority
 - Message time-to-live (TTL)
 - Message persistence
 - Transactional behavior
 - Acknowledgement policy
 - Flow control
- WLS provides a default client connection factory that:
 - Uses WebLogic's default connection settings
 - Is located on the server JNDI tree at `weblogic.jms.ConnectionFactory`



ORACLE

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

Connection Factories

Connection factories are resources that enable JMS clients to create JMS connections. A connection factory supports concurrent use, enabling multiple threads to access the object simultaneously. WebLogic JMS provides preconfigured default connection factories that can be enabled or disabled on a per-server basis. You can also configure one or more connection factories to create connections with predefined options that better suit your application.

Some connection factory options are dynamically configurable. You can modify the following parameters for connection factories:

- General configuration parameters, including modifying the default client parameters, default message delivery parameters, load-balancing parameters, unit-of-order parameters, and security parameters
- Transaction parameters, which enable you to define a value for the transaction timeout option and to indicate whether an XA queue or XA topic connection factory is returned, and whether the connection factory creates sessions that are XA aware
- Flow control parameters, which enable you to tell a JMS server or a destination to slow down message producers when it determines that it is becoming overloaded

When connection factory options are modified at run time, only the incoming messages are affected; stored messages are not affected.

Connection Factories (continued)

Within each JMS module, the connection factory resource names must be unique. All connection factory JNDI names in any JMS module must be unique across an entire WebLogic domain. Oracle WebLogic Server adds the connection factory names to the JNDI space during startup, and the application then retrieves a connection factory using the WebLogic JNDI APIs.

You can establish clusterwide, transparent access to JMS destinations from any server in the cluster, either by using the default connection factories for each server instance or by configuring one or more connection factories and targeting them to one or more server instances in the cluster. This way, each connection factory can be deployed on multiple Oracle WebLogic Server instances.

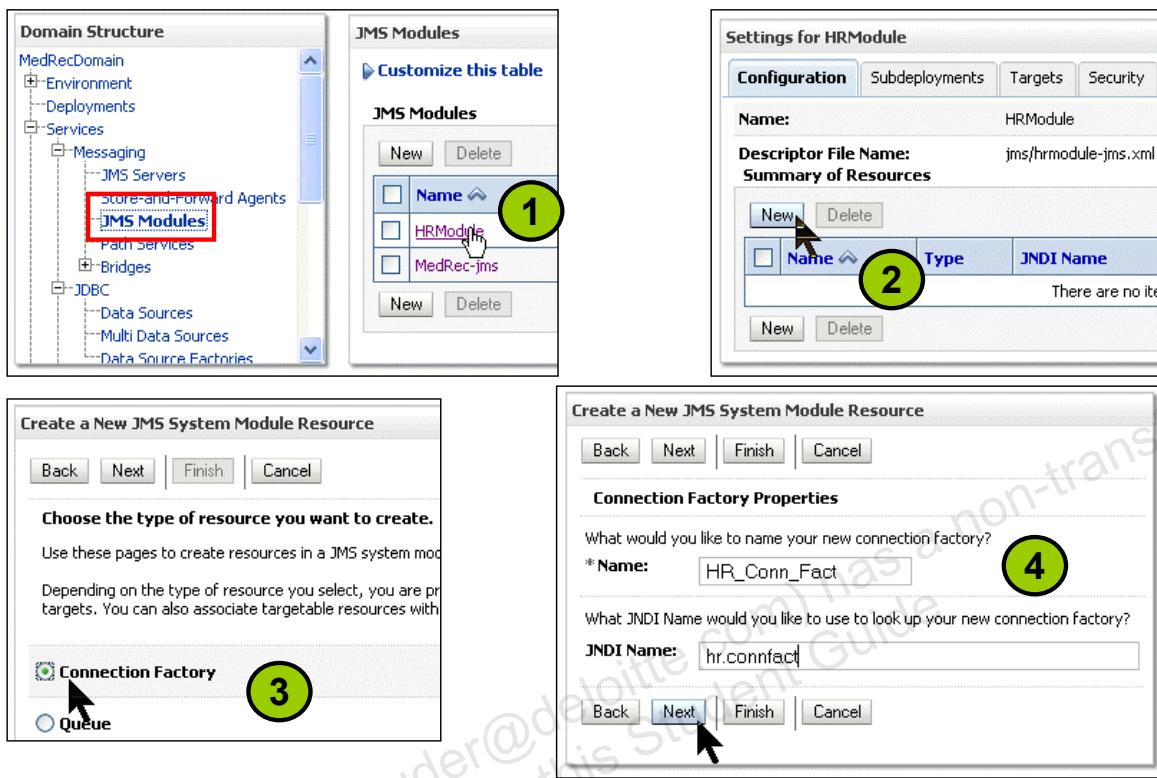
Using a Default Connection Factory

Oracle WebLogic Server defines two default connection factories, which can be looked up using the following JNDI names:

- `weblogic.jms.ConnectionFactory`
- `weblogic.jms.XAConnectionFactory`

You need to configure a new connection factory only if the preconfigured settings of the default factories are not suitable for your application. The main difference between the preconfigured settings for the default connection factories and a user-defined connection factory is the default value for the XA Connection Factory Enabled option to enable JTA transactions. Also, using default connection factories means that you have no control over targeting the Oracle WebLogic Server instances where the connection factory may be deployed. However, you can enable or disable the default connection factories on a per-Oracle WebLogic Server basis.

Creating a Connection Factory



ORACLE

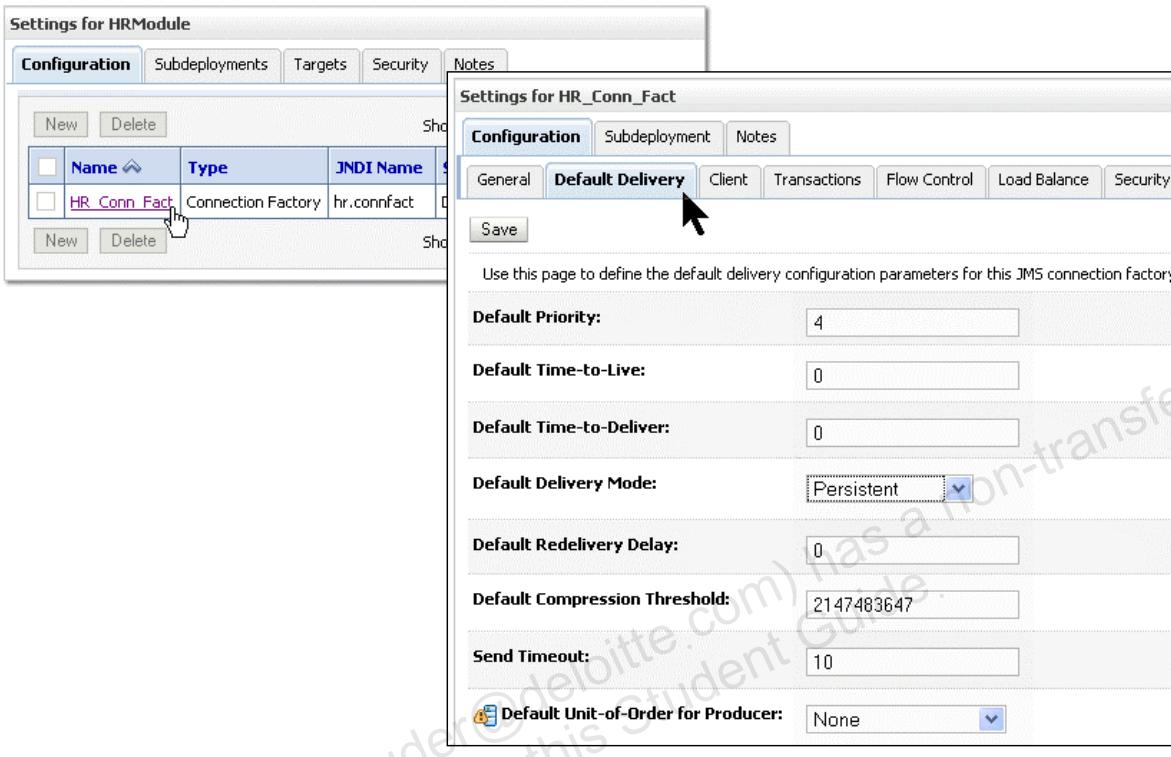
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Creating a Connection Factory

Within each JMS module, the connection factory resource names must be unique. All the connection factory JNDI names in any JMS module must be unique across an entire WebLogic domain.

1. In the Administration Console, expand Services > Messaging, and click JMS Modules. Select an existing JMS module.
2. In the “Summary of Resources” table, click New.
3. Select the Connection Factory resource type and click Next.
4. Enter Name and JNDI Name for the new connection factory, and click Next.
5. (*Not shown*) For basic default targeting, accept the default targets that are presented in the Targets box, and then click Finish. For advanced targeting, click Advanced Targeting, which allows you to select an existing subdeployment or to create a new one. When a valid subdeployment is selected, its targeted JMS servers, servers, or cluster appear as selected in the Targets box.

Configuring a Connection Factory



ORACLE

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

Configuring a Connection Factory

In the Administration Console, navigate to the connection factory resource that you want to configure. Click the Configuration tab as shown in the slide. You can configure the properties using the Default Delivery subtab.

- Default Priority:** The default priority that is used for messages when a priority is not explicitly defined. Values are between 0 and 9.
- Default Time-to-Live:** The maximum length of time, in milliseconds, that a message will exist. This value is used for messages when a priority is not explicitly defined. A value of 0 indicates that the message has an infinite amount of time to live.
- Default Time-to-Deliver:** The delay time, in milliseconds, between when a message is produced and when it is made visible on its destination
- Default Delivery Mode:** Whether or not messages should use a persistent store, if one is associated with the JMS server
- Default Redelivery Delay:** The delay time, in milliseconds, before rolled back or recovered messages are redelivered
- Send Timeout:** The maximum length of time, in milliseconds, that a sender will wait when there is not enough space available (no quota) on a destination to accommodate the message being sent. The default time is 10 milliseconds.

Destination

- A destination is a lightweight object that is stored in JNDI.
- It is the target on a JMS server for sending messages and the location from where messages will be consumed.
- The JMS destination types are:
 - Queue (for the point-to-point model)
 - Topic (for the Publish/Subscribe model)



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

Destination

A JMS destination identifies a queue (point-to-point) or topic (Publish/Subscribe) for a JMS server.

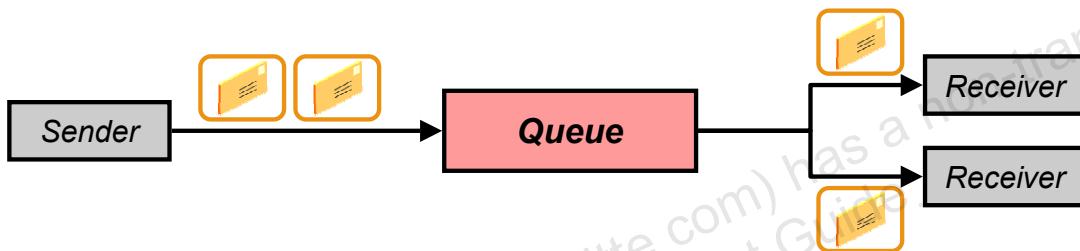
After configuring a JMS server, configure one or more queue or topic destinations for each JMS server. You configure destinations explicitly or by configuring a destination template that can be used to define multiple destinations with similar attribute settings.

A JMS destination identifies a queue (point-to-point) or topic (Publish/Subscribe) resource within a JMS module. Each queue and topic resource is targeted to a specific JMS server. A JMS server's primary responsibility for its targeted 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.

Queue Destinations

In JMS point-to-point messaging, note the following:

- Clients communicate with a queue destination.
- Messages are distributed to consumers in a serial fashion (first in, first out).
- Each message is delivered only to a single consumer.



ORACLE

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

Queue Destinations

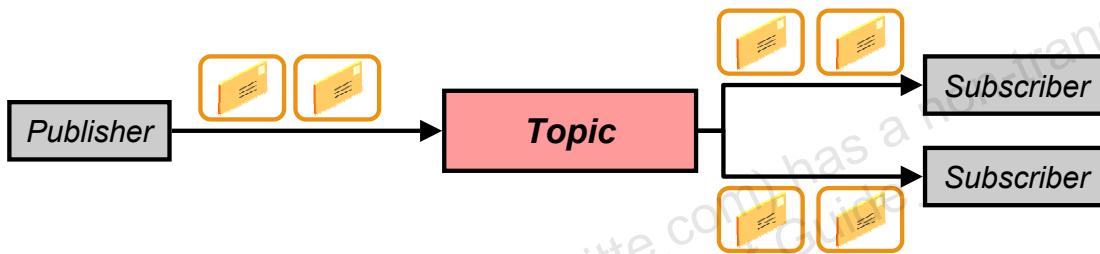
The PTP messaging model enables one application to send a message to another. PTP messaging applications send and receive messages using named queues. A queue sender (producer) sends a message to a specific queue. A queue receiver (consumer) receives messages from a specific queue. Multiple queue senders and queue receivers can be associated with a single queue, but an individual message can be delivered to only one queue receiver.

If multiple queue receivers are listening for messages on a queue, WebLogic JMS determines which one will receive the next message on a first come, first serve basis. If no queue receivers are listening on the queue, messages remain in the queue until a queue receiver attaches to the queue.

Topic Destinations

In JMS Publish/Subscribe messaging, the following is true:

- Clients communicate with a topic destination.
- Messages are broadcast to all subscribers.
- A message can be saved until at least one subscriber has consumed it (“durable”).



ORACLE

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

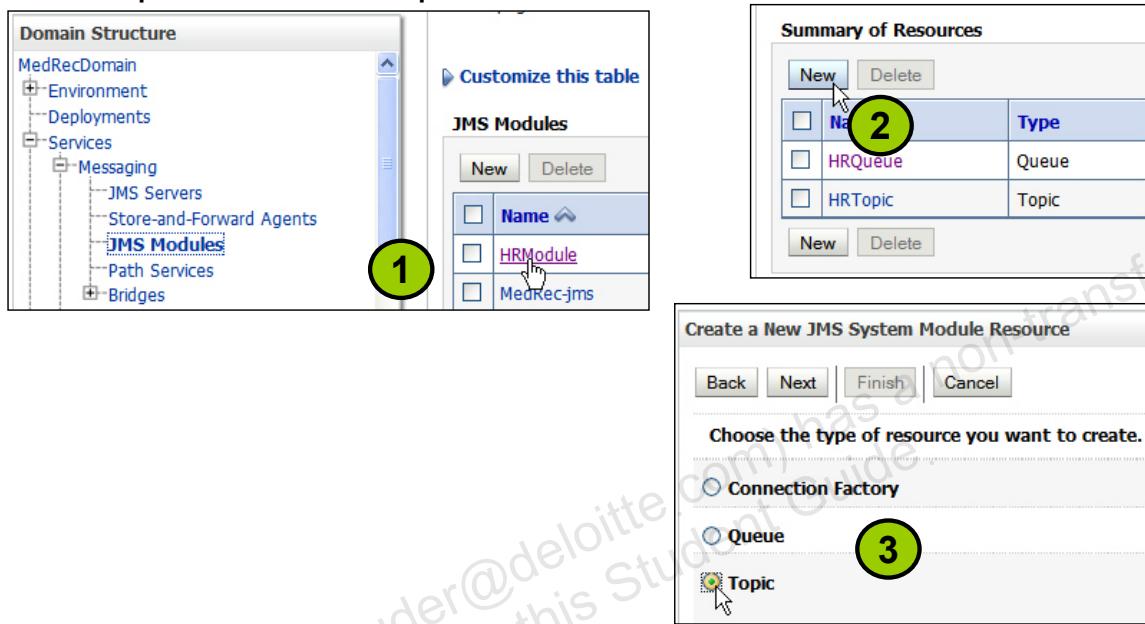
Topic Destinations

The Publish/Subscribe (pub/sub) messaging model enables an application to send a message to multiple applications. Pub/sub messaging applications send and receive messages by subscribing to a topic. A topic publisher (producer) sends messages to a specific topic. A topic subscriber (consumer) retrieves messages from a specific topic. Unlike the PTP messaging model, the pub/sub messaging model allows multiple topic subscribers to receive the same message. JMS retains the message until all topic subscribers have received it.

The pub/sub messaging model supports durable subscribers. For durable subscriptions, WebLogic JMS stores a message in a persistent file or database until the message is delivered to the subscribers or has expired, even if those subscribers are not active at the time the message is delivered. To support durable subscriptions, a client identifier (client ID) must be defined for the connection by the JMS client application. Support for durable subscriptions is a feature that is unique to the pub/sub messaging model, so client IDs are used only with topic connections; queue connections also contain client IDs, but JMS does not use them.

Creating a Destination (Topic)

- The steps to create a topic are shown here.
- Steps to create a queue are also similar.



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

ORACLE

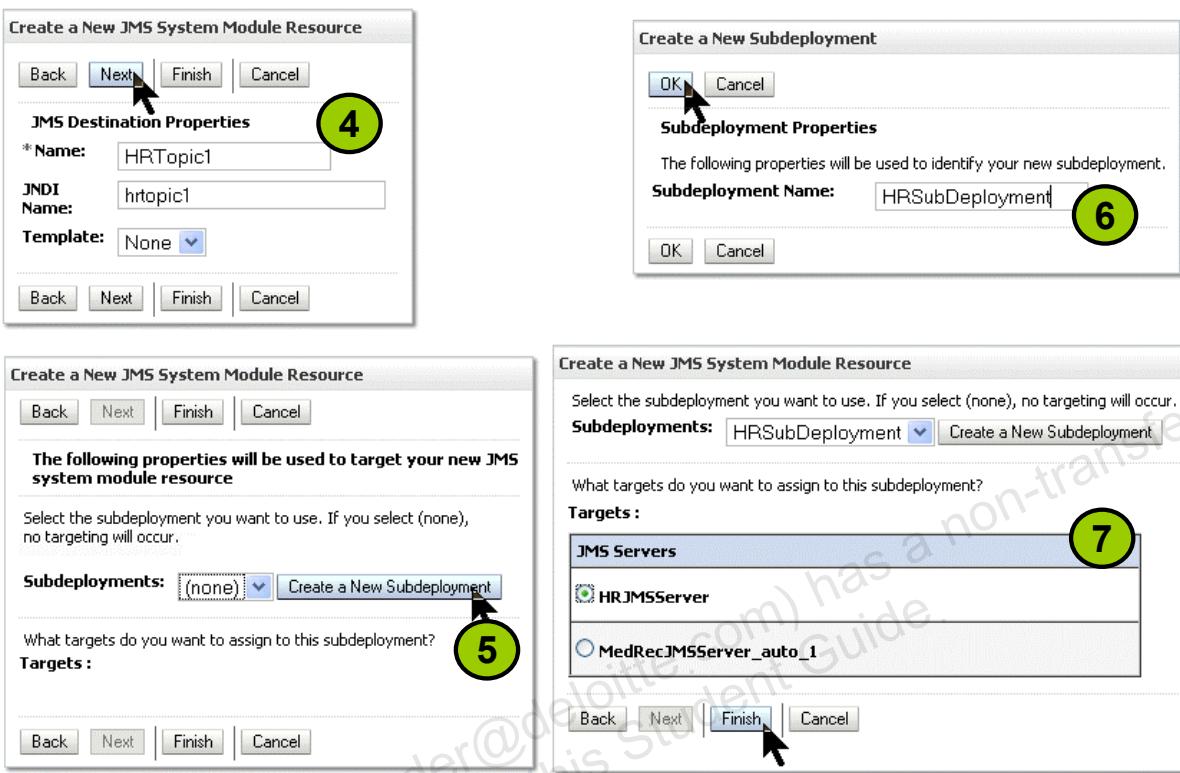
Creating a Destination (Topic)

After you create a JMS system module, you can configure resources for the module, including stand-alone queues and topics, distributed queues and topics, connection factories, JMS templates, destination sort keys, destination quota, foreign servers, and JMS store-and-forward (SAF) parameters.

For each destination, you can optionally select a subdeployment or create a new subdeployment for the resource. A subdeployment is a mechanism by which targetable JMS module resources (such as queues, topics, and connection factories) are grouped and targeted to a server resource (such as JMS servers, server instances, or a cluster).

- In the Administration Console, expand Services > Messaging and click JMS Modules. Select an existing JMS module.
- On the Configuration page, click New above the “Summary of Resources” table.
- Select the type of destination to create: Queue or Topic. Click Next.

Creating a Destination (Topic)



ORACLE

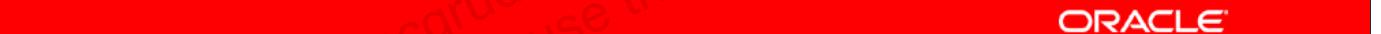
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Creating a Destination (Topic) (continued)

4. Enter Name and JNDI Name for the destination. Click Next.
5. A subdeployment is a mechanism by which JMS resources are grouped and targeted to a server instance, cluster, or SAF agent. If necessary, you can create a new subdeployment by clicking the “Create a New Subdeployment” button. You can also reconfigure subdeployment targets later by using the parent module’s subdeployment management page. If a subdeployment already exists, pick that name from the pull-down, click Next and skip to step 7.
6. Specify a subdeployment name.
7. Select an existing Subdeployment from this JMS module (such as the one you just created or one previously created). Your new JMS destination will be targeted to the JMS servers indicated by the subdeployment. If the subdeployment already exists, the targets will already be specified.

Threshold, Quota, and Paging

- Thresholds and Quotas enable you to control the size and number of message flow through JMS Servers.
- A threshold is a limit that triggers flow control and logged warnings.
- A quota is a limit defined for the JMS-administered objects; it includes the following values:
 - The maximum number of bytes that can be stored
 - The maximum number of messages that can be stored
- The message paging feature enables automatic clearing of virtual memory especially for nonpersistent messages.
- You can specify an appropriate folder structure for writing paged-out messages.

The red bar contains the word "ORACLE" in white capital letters.

ORACLE

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

Threshold and Quota

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. Flow control thresholds are attributes used for configuring size and number of message thresholds for the JMS server and its destinations. When the upper or lower threshold is reached, triggered events are launched.

Quotas limit the number of messages or the size of all the messages that can be stored. Messages sent that would put the intended target over its quota are rejected, and the sender receives an exception. Quotas enable administrators to control the size of the backlog.

Paging

With the message paging feature, JMS servers automatically attempt to free up virtual memory during peak message load periods. Message paging is always enabled on JMS servers, and so a message paging directory is automatically created without having to configure one. You can specify a directory and paged-out messages are written to files in this directory.

If a JMS server is associated with a file store (either user-defined or the server's default store), paged persistent messages are generally written to that file store, while nonpersistent messages are always written to the JMS server's paging directory.

Configuring Thresholds and Quotas



ORACLE

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

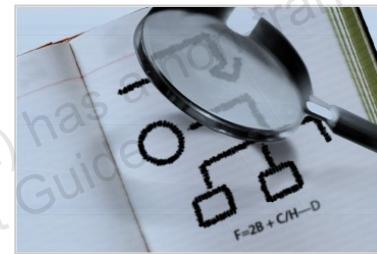
Configuring Thresholds and Quotas

After you have created either servers or destinations, you may configure their thresholds and quotas. This slide shows a sample panel for a queue. A value of –1 means that the threshold is disabled or there is no quota limit.

- **Bytes Threshold High:** The upper byte threshold beyond which specified JMS events are triggered
- **Bytes Threshold Low:** The lower byte threshold below which specified JMS events are triggered
- **Messages Threshold High:** The upper threshold to trigger events based on the number of messages stored in the destination
- **Messages Threshold Low:** The lower threshold that triggers events based on the number of messages stored in the destination
- **Bytes Maximum:** The maximum number of bytes that may be stored in this destination
- **Messages Maximum:** The maximum number of messages that may be stored in the destination
- **Blocking Send Policy:** Determines whether smaller messages are delivered before larger ones when a destination has exceeded its maximum number of messages
 - FIFO prevents the JMS server from delivering smaller messages when larger ones are already waiting for space.
 - Preemptive allows smaller send requests to preempt previous larger ones when there is sufficient space for smaller messages on the destination.

Road Map

- Oracle WebLogic Server JMS administration
- Configuring JMS objects
- Durable subscribers and persistent messaging
 - Durable subscribers
 - Configuring durable subscribers
 - Persistent and nonpersistent messages
 - Persistent backing stores using the Console
- Monitoring JMS



ORACLE

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

Durable Subscribers and Subscriptions

- Durable subscribers register durable subscriptions for guaranteed message delivery even if the subscribers are inactive.
- A subscriber is considered active if the Java object that represents it exists.
- By default, subscribers are nondurable.
- Administrators:
 - Specify where the messages are persisted
 - Configure persistent connection factories and destinations



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

Durable Subscribers and Subscriptions

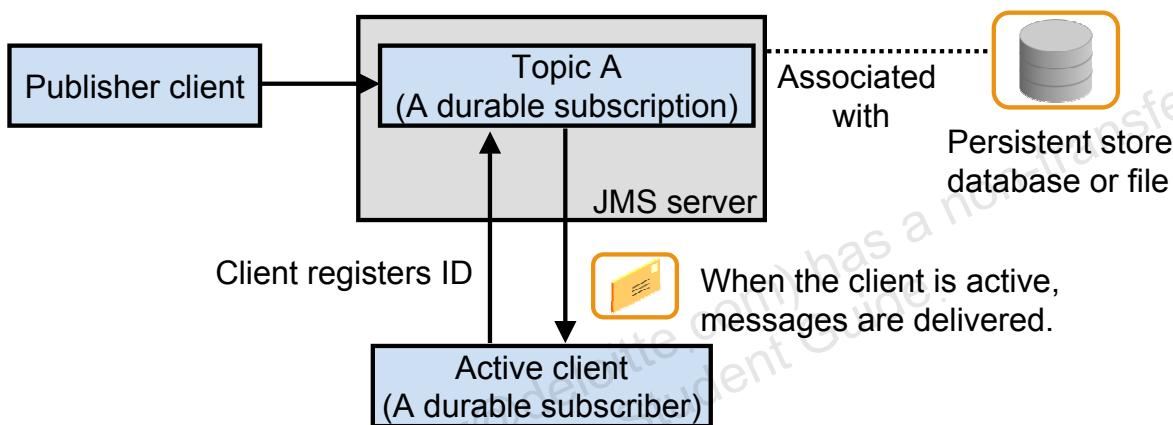
Nondurable subscriptions last for the lifetime of their subscriber object. That is, a client will see the messages published on a topic only while its subscriber is active. An inactive subscriber does not see messages that are published on its topic.

Support for durable subscriptions is a feature that is unique to the Publish/Subscribe messaging model. Client IDs are used only with topic connections.

An inactive durable subscription is one that exists but does not currently have a message consumer subscribed to it.

How a Durable Subscription Works

- Durable subscription is effective only when the client is inactive during the time that the message is published.
- When the client becomes active again, its ID is used to retrieve and redeliver messages.



ORACLE

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

How a Durable Subscription Works

A durable subscriber 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.

Sessions with durable subscribers must always provide the same client identifier. Each client must specify a name that uniquely identifies (within the client identifier) each durable subscription that it creates. Only one session at a time can have a TopicSubscriber for a particular durable subscription.

Configuring a Durable Subscription

- To configure durable subscriptions, an administrator must:
 - Create and configure a JMS store
 - Configure connection factories or destinations as persistent
 - Associate the JMS store with the JMS server
- The JMS store can be configured to use either of the following:
 - A file store
 - A JDBC Store (a connection pool)

ORACLE

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

Configuring a Durable Subscription

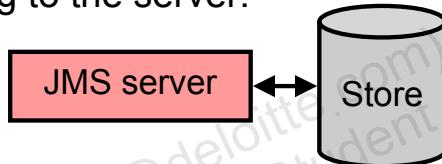
No two JMS servers can use the same backing store.

File persistence is much faster than JDBC because JDBC persistence relates to reads from and writes to a database that could potentially be a bottleneck for your system. Synchronization occurs one by one. To enhance the speed and efficiency of persisting to a database, you may like to consider the use of Oracle Coherence.

JMS backing stores can increase the amount of memory required during the initialization of an Oracle WebLogic Server as the number of stored messages increases. If initialization fails due to insufficient memory, when you are rebooting an Oracle WebLogic Server, increase the heap size of the Java Virtual Machine (JVM) proportional to the number of messages that are stored in the JMS backing store. Try to reboot again.

Persistent Messaging

- A persistent store is a physical repository for storing persistent JMS messages.
- WebLogic JMS writes persistent messages to a disk-based file or JDBC-accessible database.
- WebLogic supports guaranteed messaging using persistent stores:
 - In-progress messages can be delivered despite server restart.
 - Topic subscribers can consume missed messages despite reconnecting to the server.



ORACLE

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

Persistent Messaging

A persistent message is guaranteed to be delivered only once. 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.

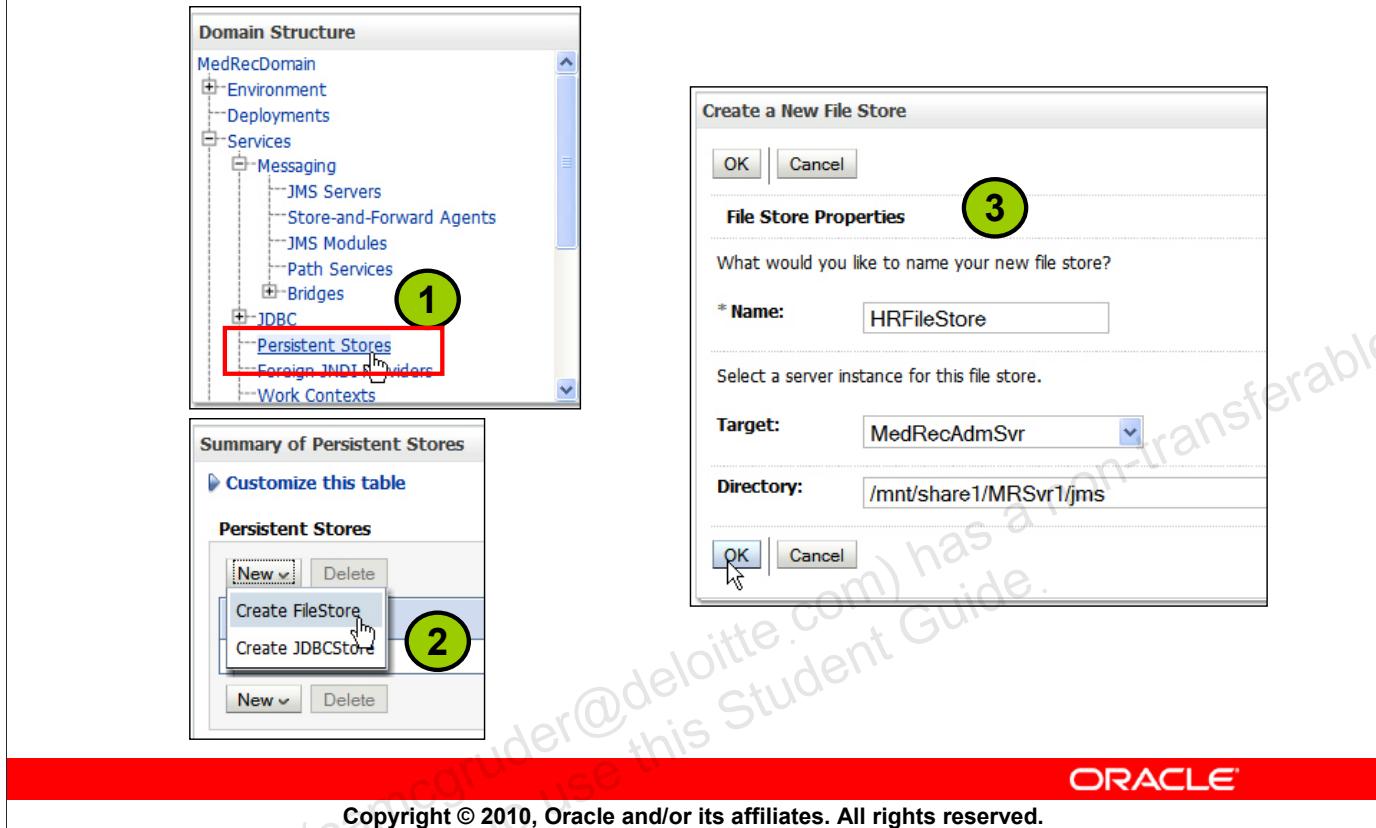
Nonpersistent messages are not stored. If a connection is closed or recovered, all nonpersistent messages that have not yet been acknowledged will be redelivered. After a nonpersistent message is acknowledged, it will not be redelivered.

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. 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 the `data\store\default` directory of a server instance.

Configure persistent messaging if:

- Development requires durable subscriptions (use durable subscribers in the application)
- You require that in-progress messages persist across server restarts

Creating a JMS Store



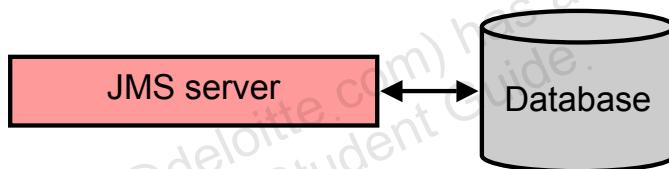
Creating a JMS Store

There is a default store for every WebLogic Server instance. The default store can be configured, but cannot point to a database. You may need to create a custom store to point to a database. Similarly, you may want to create a custom file store on your choice of storage device that can enable you to migrate the store to another server member in a cluster. When configuring a file store directory, the directory must be accessible to the server instance on which the file store is located.

1. In the left pane of the console, expand Services and select Persistent Stores.
2. On the “Summary of Persistent Stores” page, select the store type from the New list.
3. If File Store is selected, update the following on the “Create a new File Store” page:
 - **Name:** Name of the store
 - **Target:** Server instance on which to deploy the store
 - **Directory:** Path name 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.

Creating a JDBC Store for JMS

- You can create a persistent store to a database using JDBC Store.
- To configure JMS JDBC persistence, perform the following:
 - Create a JDBC DataSource.
 - Create a JDBC Store and refer to the JDBC DataSource.
 - Refer to the JMS store from the JMS server configuration.
- The required infrastructure (tables and so on) is created automatically using Data Definition Language (DDL).



ORACLE

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

Creating a JDBC Store for JMS

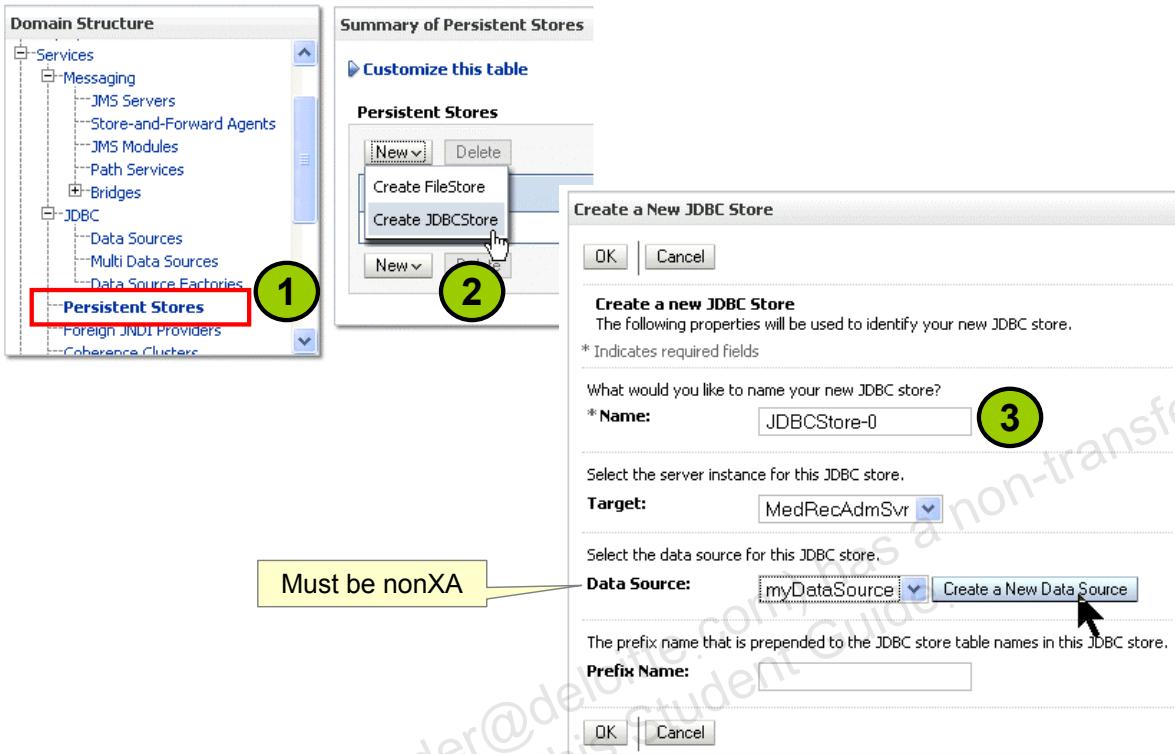
When using JMS JDBC Store, use a separate schema.

The JDBC Store Configuration page provides an optional “Create Table from DDL File” option with which you can access a preconfigured DDL file that is used to create the JDBC Store’s backing table (WLStore). This option is ignored when the JDBC Store’s backing table exists. It is mainly used to specify a custom DDL file created for an unsupported database or when upgrading the JMS JDBC Store tables from an earlier release to a current JDBC Store table.

If a DDL file name is *not* specified in the “Create Table from DDL File” field, and the JDBC Store detects that its backing table does not exist, the JDBC Store automatically creates the table by executing a preconfigured DDL file that is specific to the database vendor.

If a DDL file name is specified in the “Create Table from DDL File” field and the JDBC Store detects that its backing table does not already exist, the JDBC Store searches for the specified DDL file in the file path first, and then, if not found, searches for the DDL file in the CLASSPATH. After it is found, the SQL within the DDL file is executed to create the JDBC Store’s backing table. If the configured file is not found and the table does not already exist, the JDBC Store fails to boot.

Creating a JMS JDBC Store



ORACLE

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

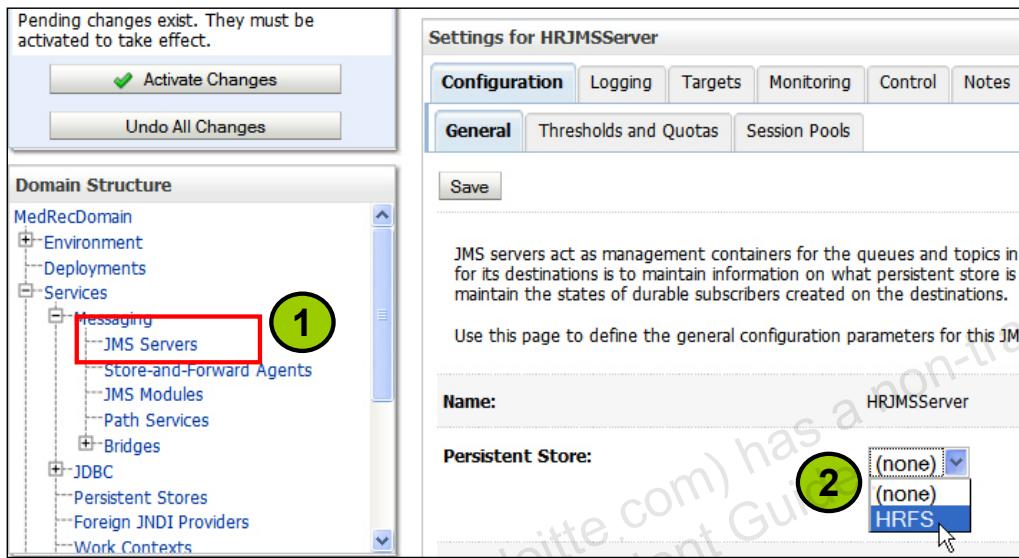
Creating a JMS JDBC Store

A Persistent Store using JDBC Store is associated with a Data Source. If a Data Source exists, you can pick it from the pull-down. If none exist (or if the existing Data Sources are XA because XA cannot be used as a JDBC Store), you must Create a New Data Source by clicking the button.

Prefix Name: The prefix for table names, if:

- The database management system requires fully qualified names, such as schema
- You must differentiate between the JMS tables for two Oracle WebLogic Servers to store multiple tables on one DBMS

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 Configuration > General tab. If this field is set to “(none),” the JMS server uses the default file store that is automatically configured on each targeted server instance.

Persistent Connection Factory

The screenshot shows the 'Settings for HRCF' configuration page. The 'Default Delivery' tab is selected. The page includes a note about defining default delivery parameters and contains the following fields:

Parameter	Value
Default Priority	4
Default Time-to-Live	0
Default Time-to-Deliver	0
Default Delivery Mode	Persistent
Default Redelivery Delay	0
Default Compression Threshold	2147483647
Send Timeout	10

ORACLE

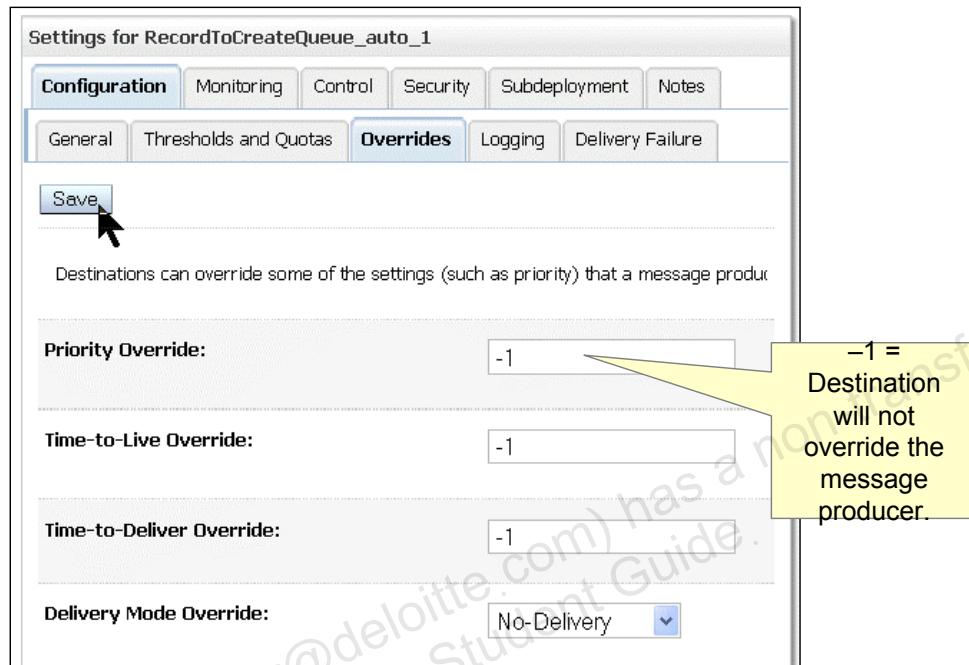
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Persistent Connection Factory

Default Delivery Mode: Used for messages for which a delivery mode is not explicitly defined. It can be persistent or nonpersistent.

The preferred method, according to the JMS specification, is to configure the connection factory with the client ID. For Oracle WebLogic Server JMS, this means adding a separate connection factory definition during configuration for each client ID. Applications look up their own topic connection factories in JNDI and use them to create connections containing their own client IDs. Alternatively, an application can set its client ID programmatically.

Configuring Destination Overrides



ORACLE

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

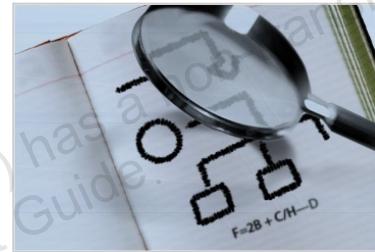
Configuring Destination Overrides

The delivery mode assigned to all messages that arrive at the destination can be set to override the delivery mode specified by the message producer. 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.

Road Map

- Oracle WebLogic Server JMS administration
- Configuring JMS objects
- Durable subscribers and persistent messaging
- Monitoring JMS
 - Monitoring JMS servers
 - Monitoring JMS modules



ORACLE

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

Monitoring JMS Servers

Statistics are provided for the following JMS objects:

- JMS servers
- Connections
- Destinations

The screenshot shows a table titled "Statistics(Filtered - More Columns Exist)". The table has a header row with columns: Name, Messages Current, Messages High, Bytes Current, and Session Pools Current. There is one data row for "PayrollJMSServer".

	Name	Messages Current	Messages High	Bytes Current	Session Pools Current
<input type="checkbox"/>	PayrollJMSServer	3	3	36	0

ORACLE

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

Monitoring JMS Servers

You can monitor the run-time statistics for an active JMS server. From the Monitoring tab, you can also access run-time information for a JMS server's destinations, transactions, connections, and server session pools.

1. Expand Services > Messaging and click JMS Servers. Select a JMS server.
2. Click the Monitoring tab. By default, a Monitoring subtab is displayed, which provides general statistics for all destinations on every JMS server in the domain. These statistics include the number and size of messages processed by the JMS server.

The Active Destinations tab displays the statistics for each active JMS destination for the domain.

The Active Transactions tab displays all active JMS transactions for the domain. For troubleshooting, you can force commits or rollbacks on selected transactions. Simply select a transaction, and then click either the Force Commit or the Force Rollback button.

The Active Connections tab displays all active client JMS connections for the domain. For troubleshooting, you can destroy selected connections. Simply select a connection, and then click the Destroy button above the table.

Monitoring and Managing Destinations (Active Queues and Topics)

This page allows you to view active destinations targeted to this JMS server.

Customize this table

Show fewer or additional data points on this page by expanding “Customize this table” and modifying the Column Display.

	Name	Pause Resume	Msgs Current	Msgs Pending	Msgs High	Msgs Recv'd	Msgs Threshold	Destination Type	State	Production Paused	Insertion Paused	Consumption Paused
<input type="checkbox"/>	MedRec-jms!PatientNotificationQueue_auto_1		0	0	0	0	0	Queue	advertised_in_cluster_jndi	false	false	false
<input type="checkbox"/>	MedRec-jms!RecordToCreateQueue_auto_1		0	0	0	0	0	Queue	advertised_in_cluster_jndi	false	false	false
<input type="checkbox"/>	MedRec-jms!WSRMDefaultQueue_auto_1		0	0	0	0	0	Queue	advertised_in_cluster_jndi	false	false	false

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

ORACLE

Monitoring and Managing Destinations (Active Queues and Topics)

You can use this page to monitor information about a JMS consumer, which receives messages from a JMS queue (QueueReceiver) or topic (TopicSubscriber).

For troubleshooting, you can temporarily pause all run-time message production, insertion (in-flight messages), and consumption operations on any or all destinations targeted to the selected JMS server. These “message pausing” options enable you to assert administrative control over the JMS subsystem behavior in the event of an external resource failure.

The available columns include:

- **Messages Current:** The current number of messages in the destination. This does not include the pending messages.
- **Messages Pending:** The number of pending messages in the destination. A pending message is one that has either been sent in a transaction and not been committed or that has been received and not been committed or acknowledged.
- **Messages High:** The peak number of messages in the destination since the last reset
- **Messages Received:** The number of messages received in this destination since the last reset
- **Messages Threshold:** The amount of time in the threshold condition since the last reset
- **Consumers Current:** The current number of consumers accessing this destination

Monitoring Queues

- In the Administration console, select Services > Messaging > JMS Modules.
- In the JMS Modules table, click the JMS module you have created.
- In the “Summary of Resources” table, click the link to your queue, and then click the Monitoring tab.
- The Messages High and Messages Total columns show nonzero values indicating that messages have been received.

Destinations (Filtered - More Columns Exist)						
		Showing 1 to 1 of 1 Previous Next				
	Name	Consumers Current	Consumers High	Consumers Total	Messages High	Messages Total
<input type="checkbox"/>	dizzyworldModuleIdizzyworldQueue	0	0	0	1	1
Show Messages		Showing 1 to 1 of 1 Previous Next				

ORACLE

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

Monitoring Queues

Use this page to view run-time statistics about the current queue resource. Run-time statistics include counts, pending, and threshold data for consumers, bytes, and messages for the queue.

To access the queue’s message management page, select the check box next to its name, and then click the Show Messages button.

Managing Messages in a Queue

- You can enable messages to be viewed in the Administration Console.
- After they are enabled, you can view and manage the messages in a queue using the Administration Console.

The screenshot shows the 'Settings for HRQueue' page in the Administration Console. The 'Monitoring' tab is selected. On the left, there's a 'Customize this table' section for 'Destinations'. A 'Show Messages' button is highlighted with a mouse cursor. On the right, the 'JMS Messages(Filtered - More Columns Exist)' table is displayed, showing three rows of message details:

ID	CorrId	Time Stamp
ID:<28135.1238368149351.0>		Sun Mar 29 19:09:09 EDT 2009
ID:<28135.1238368149351.0>		Sun Mar 29 19:09:09 EDT 2009

Destinations(Filtered - More Columns Exist)

Name	Messages Current	Messages Pending	Messages Total	Consumers Current
HRModule!HRQueue	2	0	0	0

ORACLE

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

Managing Messages in a Queue

You can enable viewing of messages in the Administration Console using these steps:

1. In the Administration Console, navigate to the queue resource that you want to configure:
 - Navigate to JMS Resources in System Modules, and then to JMS resources in an application module
2. Click the Monitoring tab.
3. Select the check box next to the queue, and then click Show Messages to access the queue's JMS Messages table.
4. You can then perform the following administrative procedures on a specific message or selected messages:
 - Click a message in the queue to open the View Contents page, where you can view the contents of a JMS message.
 - Click New to create a new JMS message. You can specify header and body content when creating the message, which will then be produced on the current queue.
 - Select messages for deletion and click Delete to delete them from the current queue.
 - Click Move to transfer selected JMS messages from the current queue to another destination, including a destination on a different JMS server.

Quiz

Which are the correct messaging model and JMS destination type associations?

- a. Queue: Publish/Subscribe
- b. Queue: Point-to-point
- c. Topic: Publish/Subscribe
- d. Topic: Point-to-point



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

Quiz

Which are the available resource types within an Oracle WebLogic Server JMS module?

- a. Connection factory
- b. Queue
- c. Topic
- d. Server
- e. Store



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

Summary

In this lesson, you should have learned how to:

- Describe how Oracle WebLogic Server JMS is implemented
- Configure JMS server
- Configure connection factories
- Configure queues and topics
- Configure persistent messages
- Monitor JMS resources and messages



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

Practice 15 Overview: Configuring JMS Resources

This practice covers the following topics:

- Configuring JMS resources such as:
 - JMS server, JMS module, queue, and topic
- Posting messages to the queue and topic
- Monitoring a queue in the Administration Console



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

16

Introduction to Clustering

ORACLE®

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

Objectives

After completing this lesson, you should be able to do describe:

- The benefits of Oracle WebLogic cluster
- Basic cluster architecture
- Multitier cluster architecture
- Communication among clustered server instances
- The key criteria for selecting suitable cluster architecture



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

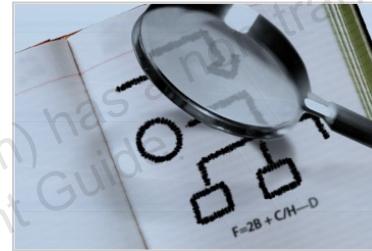
Objectives

Scenario

Clustering provides availability and scalability benefits to servers. As the administrator at MedRec, you want to understand the benefits of clustering and the architectural considerations to help you decide on the appropriate structure for your environment.

Road Map

- Oracle WebLogic cluster introduction
 - What is a cluster?
 - Benefits of clustering
 - HTTP clustering and proxy plug-in
 - Introduce EJB clustering
- Cluster architecture
- Cluster communication



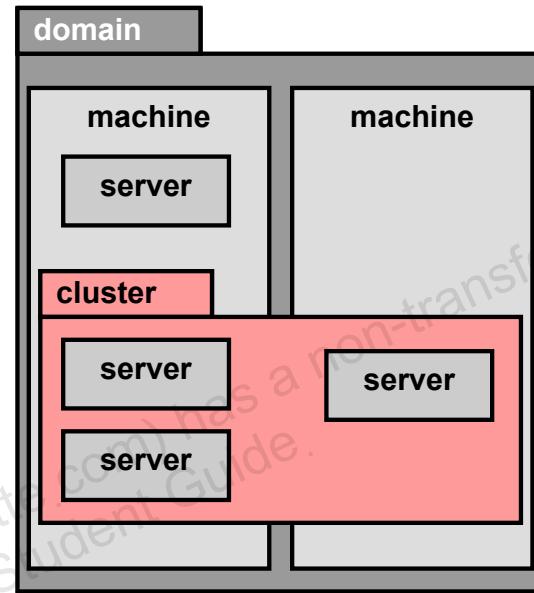
ORACLE

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

What Is a Cluster?

A cluster:

- Is a logical group of managed servers within a domain
- Supports features to provide high availability for:
 - Whole servers
 - Web applications and services
 - EJB applications
 - JDBC resources
 - JMS
- Is transparent to clients



ORACLE

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

What Is a Cluster?

An Oracle WebLogic Server cluster consists of one or more Oracle WebLogic Server instances running simultaneously and working together to provide increased scalability and reliability. A cluster appears to clients as one Oracle WebLogic Server instance. The server instances that constitute a cluster can run on one machine or on different machines.

By replicating the services provided by one instance, an enterprise system achieves a fail-safe and scalable environment. It is good practice to set all the servers in a cluster to provide the same services.

You can increase a cluster's capacity by adding server instances to the cluster on an existing machine, or by adding machines to the cluster to host the incremental server instances.

The clustering support for different types of applications is as follows:

- For Web applications, the cluster architecture enables 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 an external load-balancer.
- For Enterprise JavaBeans (EJBs) and Remote Method Invocation (RMI) objects, clustering uses 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.
- For JMS applications, clustering supports clusterwide transparent access to destinations from any member of the cluster.

Benefits of Clustering

Concept	Description
Scalability	It provides more capacity for an application by adding servers, without having to make major architectural changes.
Load balancing	It distributes work (client requests and so on) across the members of a cluster.
Application failover	When an object in an application that is performing a task becomes unavailable, the object from the application in another server takes over to finish the job.
Availability	After a system failure on one server, it automatically continues ongoing work on another server.
Migration	After a system failure on one server, it continues ongoing work by moving the component to another server.

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

Benefits of Clustering

An Oracle WebLogic Server cluster provides the following benefits:

- **Scalability:** The capacity of a cluster is not limited to one server or one machine. Servers can be added to the cluster dynamically to increase capacity. If more hardware is needed, a new server on a new machine can be added.
- **Load Balancing:** The distribution of jobs and associated communications across the computing and networking resources in your environment can be even or weighted, depending on your environment. Even distributions include round-robin and random algorithms.
- **Application Failover:** Distribution of applications and their objects on multiple servers enables easier failover of the session-enabled applications.
- **Availability:** A cluster uses the redundancy of multiple servers to insulate clients from failures. The same service can be provided on multiple servers in a cluster. If one server fails, another can take over. The capability to execute failover from a failed server to a functioning server increases the availability of the application to clients.
- **Migration:** This ensures uninterrupted availability of pinned services or components—those that must run only on a single server instance at any given time, such as the Java Transaction API (JTA) transaction recovery system, when the hosting server instance fails.

Understanding the technical infrastructure that enables clustering helps programmers and administrators to maximize the scalability and availability of their applications.

What Can Be Clustered

The following types of objects can be clustered:

- Servlets
- JSP
- EJB
- RMI objects
- Java Message Service (JMS) destinations
- Java Database Connectivity (JDBC) connections



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

What Can Be Clustered

WebLogic Server provides clustering support for servlets and JSPs by replicating the HTTP session state of clients that access clustered servlets and JSPs. WebLogic Server can maintain HTTP session states in memory, a file system, or a database.

Load balancing and failover for EJBs and RMI objects are handled using replica-aware stubs, which can locate instances of the object throughout the cluster. Replica-aware stubs are created for EJBs and RMI objects as a result of the object compilation process. EJBs and RMI objects are deployed homogeneously to all the server instances in the cluster.

WebLogic JMS architecture implements clustering of multiple JMS servers by supporting clusterwide, transparent access to destinations from any WebLogic Server instance in the cluster.

WebLogic Server enables you to cluster JDBC objects, including data sources and multidata sources, to improve the availability of cluster-hosted applications. Each JDBC object that you configure for your cluster must exist on each managed server in the cluster—when you configure the JDBC objects, target them to the cluster.

Proxy Servers for HTTP Clusters

- Proxy servers are used to provide load balancing and failover for a cluster. They:
 - Are the client's first level of interaction with the cluster
 - Give the cluster its single server appearance
- A proxy server can be either software based or hardware based.
- A software-based proxy server may be a WebLogic servlet, Web server plug-in, or a third-party application.
- A hardware-based proxy server is typically a physical load balancer.

ORACLE

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

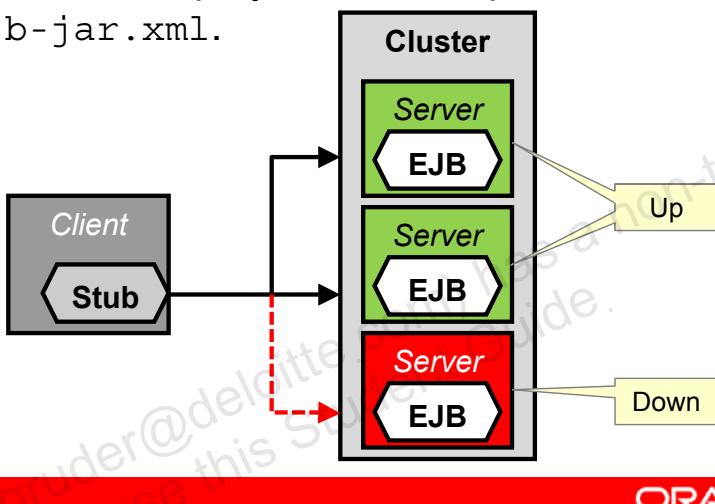
Proxy Servers for HTTP Clusters

Proxies are how clients interact with the cluster, whether they are hardware or software based. You have three basic choices of proxy depending on your cluster architecture: `HTTPClusterServlet`, a Web server plug-in, or a physical load balancer (such as Local Director or F5 Networks Big IP). These proxy choices are generally available regardless of the architecture type, but some architectures might dictate the type of proxy that will be needed. You can configure Oracle WebLogic Server clusters to operate alongside existing Web servers. In such an architecture, a bank of Web servers provides static HTTP content for the Web application, using a WebLogic proxy plug-in or `HttpClusterServlet` to direct servlet and JSP requests to a cluster.

There are two alternative proxy architectures: two-tier and multitier.

High Availability for EJBs

- WebLogic provides the EJB client applications with cluster-aware stubs that transparently perform load balancing and failover.
- You can enable and configure clustering for each EJB using the application deployment descriptor `weblogic-ejb-jar.xml`.



ORACLE

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

High Availability for EJBs

Failover for clustered EJBs 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.

With clustered objects, automatic failover generally occurs only in cases where the object is idempotent. An object is idempotent if any method can be called multiple times with no different effect than calling the method once.

Method-level failover for a stateful service requires state replication. Oracle WebLogic Server satisfies this requirement by replicating the state of the primary bean instance to a secondary server instance, using a replication scheme similar to that used for HTTP session state.

Oracle WebLogic Server uses the round-robin algorithm as the default load-balancing strategy for clustered object stubs when no algorithm is specified. Weight-based load balancing improves on the round-robin algorithm by taking into account a preassigned weight for each server. Oracle WebLogic Server provides server affinity that can be used to turn off load balancing for external client connections. If an object is configured for server affinity, the client-side stub attempts to choose a server instance to which it is already connected and continues to use the same server instance for method calls.

Road Map

- Oracle WebLogic cluster introduction
- Cluster architecture
 - Considerations for selecting an appropriate cluster architecture
 - Basic cluster architecture
 - Multitier cluster architecture
 - Proxy servers
- Cluster communication



ORACLE

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

Selecting a Cluster Architecture

- Consider the following factors when selecting a suitable architecture:
 - Performance
 - Efficient state persistence
 - Optimal load balancing
 - Effective failover
 - Reliable communication
- There are two primary cluster architectures to choose from:
 - Basic cluster architecture
 - Multitier architecture

ORACLE

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

Selecting a Cluster Architecture

Although architecture is considered subjective and good architecture is usually a point of debate, there are some general considerations that should be addressed when selecting a cluster architecture:

- Performance
- Efficient state persistence (through replication or other means)
- Optimal load balancing
- Effective failover
- Reliable communication

These factors ultimately decide the success or failure of your clustered services.

Cluster Architecture

- Applications are generally deployed in multiple tiers, each tier representing a distinct functionality:
 - Web tier
 - Presentation tier
 - Business or object tier
- WebLogic provides clustering support for all three tiers.
- Other services, such as JMS and JDBC, can take advantage of clusters. The load balancing and failover operations for these services are handled differently.

ORACLE

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

Cluster Architecture

Applications are usually broken into three functional tiers: Web tier, presentation tier, and object tier. In programming circles, these are also known as the model, view, and control. You tend to abstract them a little more when talking about clustering, but they are effectively the same.

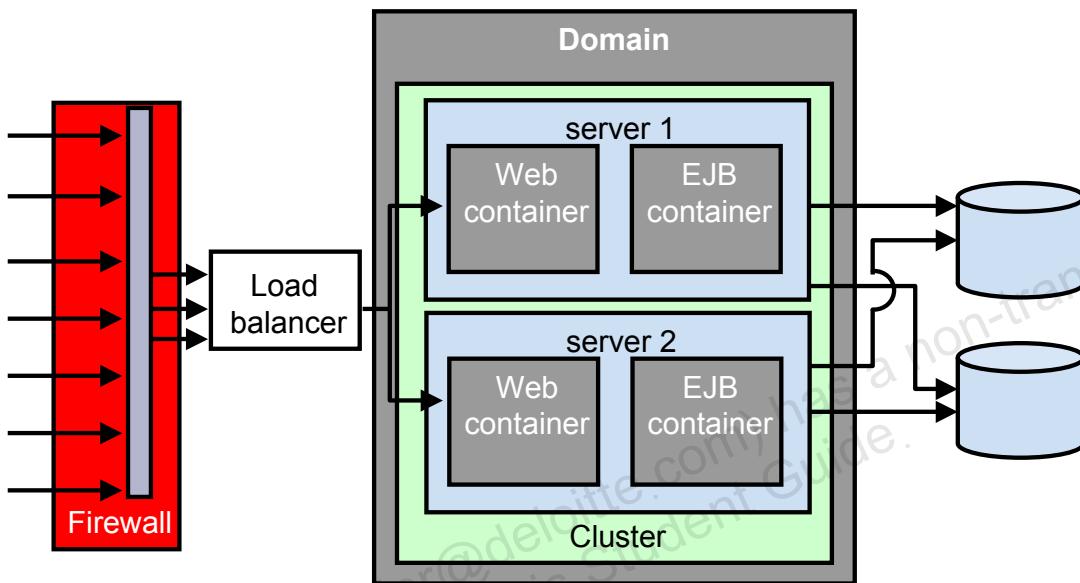
The Web tier provides the static, idempotent presentation of a Web application and is generally the first piece that clients come in contact with. Often, the Web tier is handled by a Web server, such as Oracle HTTP Server, Apache, Internet Information Server (IIS), or Netscape Enterprise Server (NES).

The presentation tier provides the dynamic content, such as servlets, JSP, and so forth. This tier also acts as a consumer to the business logic represented in the business tier. The presentation tier typically contains implemented design patterns or run-time frameworks that allow the client to interact with the business tier and generate a dynamic view of that tier per request or session. The presentation tier is handled by WebLogic and is accessed via direct or indirect client requests to the presentation tier elements.

The business tier provides access to business logic, middleware, and integrated systems. Typically, these are handled by various types of EJBs or server services, such as JMS and JDBC. WebLogic also handles this tier, but there are other applications, services, and servers that participate at this level.

Basic Cluster Architecture

A basic cluster architecture combines static HTTP, presentation logic, business logic, and objects into one cluster.



ORACLE

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

Basic Cluster Architecture

The basic recommended cluster architecture combines all Web application tiers and puts the related services (static HTTP, presentation logic, and objects) into one cluster.

The basic architecture has the following advantages:

- **Easy administration:** Because one cluster hosts static HTTP pages, servlets, and EJBs, you can configure the entire Web application and deploy or undeploy objects using one Administration Console. You do not need to maintain a separate bank of Web servers (and configure Oracle WebLogic Server proxy plug-ins) to benefit from clustered servlets.
- **Flexible load balancing:** Using load-balancing hardware directly, in front of the Oracle WebLogic Server cluster, enables you to use advanced load-balancing policies for access to both HTML and servlet content.
- **Robust security:** Putting a firewall in front of your load-balancing hardware enables you to set up a demilitarized zone (DMZ) for your Web application using minimal firewall policies.
- **Optimal performance:** The combined-tier architecture offers the best performance for applications in which most or all the servlets or JSPs in the presentation tier typically access objects in the object tier, such as EJBs or JDBC objects.

A DMZ is a logical collection of hardware and services that is made available to outside, untrusted sources.

Basic Cluster Architecture: Advantages and Disadvantages

- Advantages:
 - Easy administration
 - Flexible load balancing
 - Robust security
- Disadvantages:
 - It cannot load-balance EJB method calls.
 - Load-balancing across the tiers may become unbalanced.



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

Basic Cluster Architecture: Advantages and Disadvantages

Load balancing and failover can be introduced only at the interfaces between Web application tiers. So, when tiers are deployed to a single cluster, you can load-balance only between clients and the cluster. Because most load balancing and failover occur between clients and the cluster itself, a combined-tier architecture meets the needs of most Web applications.

However, such basic clusters provide no opportunity for load-balancing method calls to clustered EJBs. Because clustered objects are deployed on all Oracle WebLogic Server instances in the cluster, each object instance is available locally. Oracle WebLogic Server optimizes method calls to clustered EJBs by always selecting the local object instance, rather than distributing requests to remote objects.

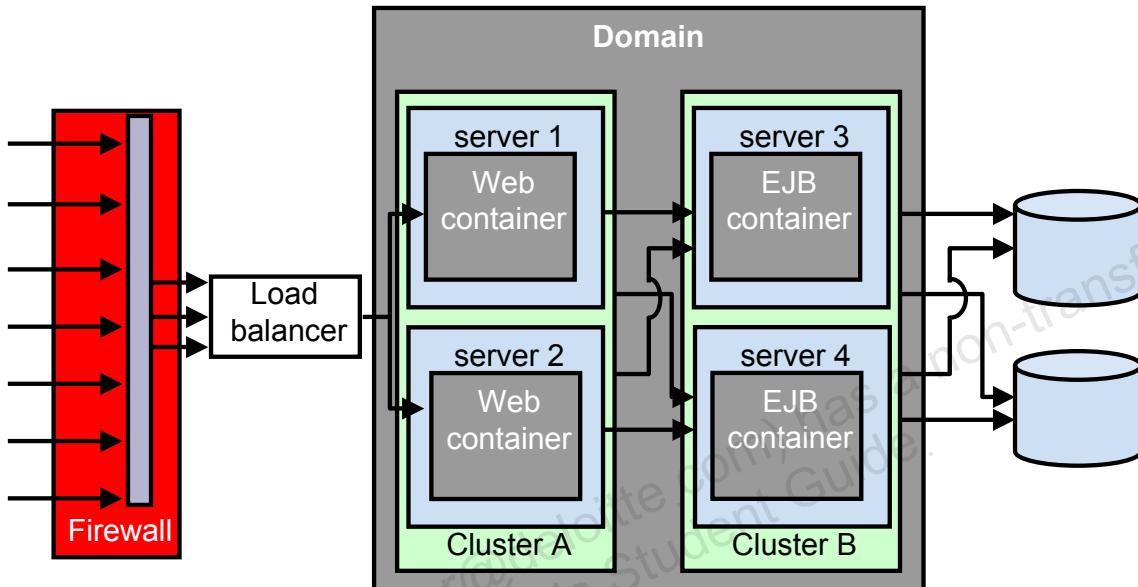
If the processing load on individual servers becomes unbalanced, it may eventually become more efficient to submit method calls to remote objects rather than process methods locally.

To use load balancing for method calls to clustered EJBs, you must split the presentation and object tiers of the Web application onto separate physical clusters, thereby ensuring that all the object calls are remote calls and the load is balanced.

Consider the frequency of invocations of the object tier by the presentation tier when you decide between a combined-tier and a multtier architecture. If presentation objects usually invoke the object tier, a combined-tier architecture may offer better performance than a multtier architecture.

Multitier Cluster Architecture

The Web tier and the business logic with services can be separated into two clusters.



ORACLE

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

Multitier Cluster Architecture

In the architecture illustrated in the slide, two separate Oracle WebLogic Server clusters are configured:

- Cluster A to serve static HTTP content and clustered servlets
- Cluster B to serve clustered EJBs

Multitier architecture is recommended for applications that require:

- Load balancing for method calls to clustered EJBs
- Flexibility for load balancing between servers that provide HTTP content and servers that provide clustered objects
- Higher availability (fewer single points of failure)
- More flexible security

Note: Consider the frequency of invocations from the presentation tier to the object tier when considering a multitier architecture. If presentation objects usually invoke the object tier, a combined-tier architecture may offer better performance than a multitier architecture.

Multitier: Advantages and Disadvantages

- Advantages:
 - Improved load balancing
 - Load balancing of EJB methods
 - Higher availability
 - Improved security options
- Disadvantages:
 - Can create a bottleneck when the presentation tier makes frequent calls to the business logic
 - Can lead to increased licensing cost
 - Can lead to added firewall configuration complexity



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

Multitier: Advantages and Disadvantages

The multitier architecture provides the following advantages:

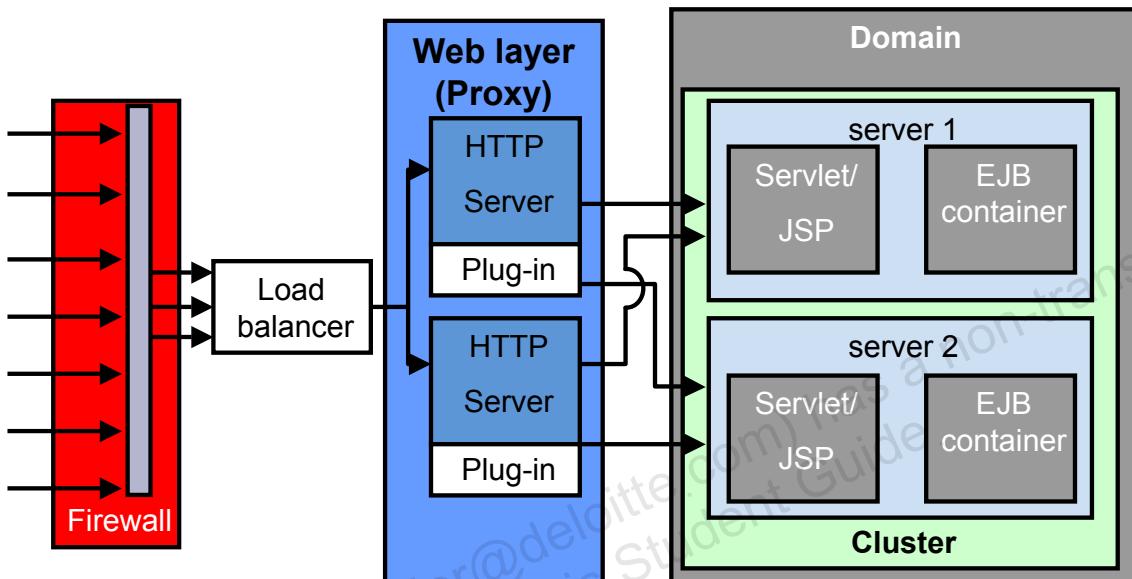
- **Load-balancing EJB methods:** By hosting servlets and EJBs on separate clusters, the servlet-method calls to the EJBs can be load-balanced across multiple servers.
- **Improved server load balancing:** Separating the presentation and object tiers onto separate clusters provides you with more options for distributing the load of the Web application. For example, if the application accesses HTTP and servlet content more often than EJB content, you can use a large number of Oracle WebLogic Server instances in the presentation tier cluster to concentrate access to a smaller number of servers that host the EJBs. For example, if your Web clients make heavy use of servlets and JSPs but access a relatively small set of clustered objects, the multitier architecture enables you to concentrate the load of servlets and EJB objects appropriately. You may configure a servlet cluster of 10 Oracle WebLogic Server instances and an object cluster of three managed servers, while still fully using each server's processing power.
- **Higher availability:** By using additional Oracle WebLogic Server instances, the multitier architecture has fewer points of failure than the basic cluster architecture. For example, if an Oracle WebLogic Server that hosts the EJBs fails, the HTTP- and servlet-hosting capacity of the Web application is not affected.

Multitier: Advantages and Disadvantages (continued)

- **Improved security options:** By separating the presentation and object tiers onto separate clusters, you can use a firewall policy that places only the servlet/JSP cluster in the DMZ. Servers that host the clustered objects can be further protected by denying direct access from untrusted clients.
- **Limitations of Multitier Architectures**
 - **No Collocation Optimization:** The multitier architecture cannot optimize object calls by using the collocation strategy. So, the Web application may incur network overhead for all method calls to the clustered objects.
 - **Firewall Restrictions:** If you place a firewall between the servlet cluster and an object cluster in a multitier architecture, you must bind all the servers in the object cluster to public DNS names, rather than IP addresses. Binding those servers with IP addresses can cause address translation problems and prevent the servlet cluster from accessing individual server instances.

Basic Cluster Proxy Architecture

This is similar to the basic cluster architecture, except that static content is hosted on HTTP servers.



ORACLE

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

Basic Cluster Proxy Architecture

The two-tier proxy architecture contains two physical layers of hardware and software.

Web Layer

The proxy architecture uses a layer of hardware and software that is dedicated to the task of providing the application's Web tier. This physical Web layer can consist of one or more identically configured machines that host one of the following application combinations:

- Oracle WebLogic Server with `HttpClusterServlet`
- Apache with the Oracle WebLogic Server Apache proxy plug-in
- Netscape Enterprise Server with the Oracle WebLogic Server NSAPI proxy plug-in
- Microsoft Internet Information Server with the Oracle WebLogic Server Microsoft-IIS proxy plug-in

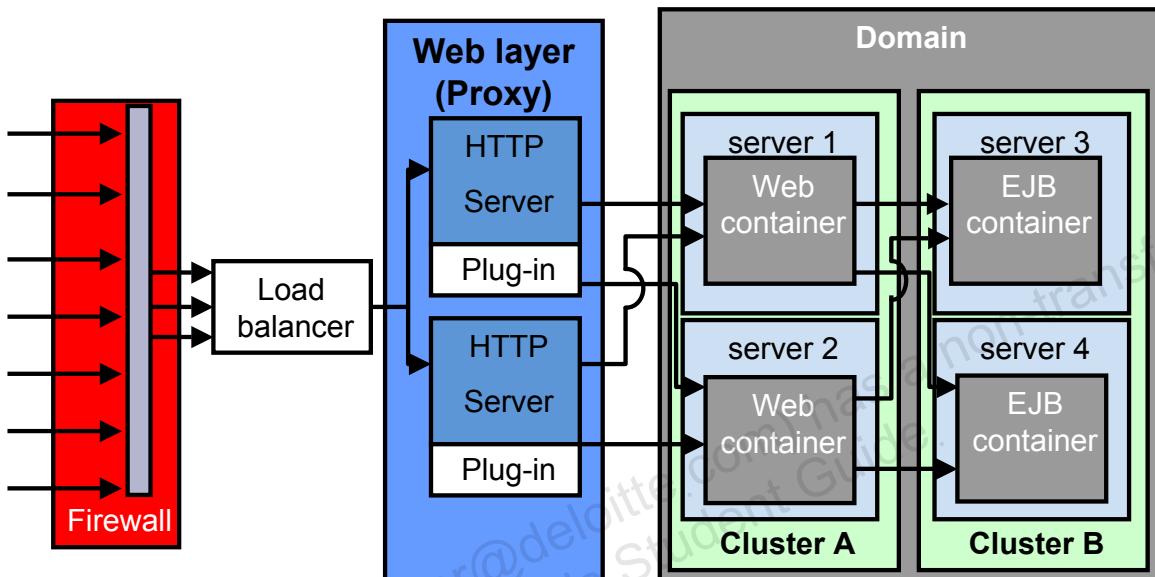
Regardless of which Web server software you select, remember that the physical tier of the Web servers should provide only static Web pages. Dynamic content—servlets and JSPs—are proxied via the proxy plug-in or `HttpClusterServlet` to an Oracle WebLogic Server cluster that hosts servlets and JSPs for the presentation tier.

Servlet/Object Layer

The recommended two-tier proxy architecture hosts the presentation and object tiers on a cluster of Oracle WebLogic Server instances. This cluster can be deployed either on a single machine or on multiple separate machines. The Servlet/Object layer differs from the combined-tier cluster in that it does not provide static HTTP content to application clients.

Multitier Cluster Proxy Architecture

This is similar to the multitier cluster architecture, except that static content is hosted on HTTP servers.



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

ORACLE

Multitier Cluster Proxy Architecture

You can also use a bank of Web servers as the front end to a pair of Oracle WebLogic Server clusters that host the presentation and object tiers.

Using stand-alone Web servers and proxy plug-ins provides the following advantages:

- You can use existing hardware.
- If you already have a Web application architecture that provides static HTTP content to clients, you can easily integrate the existing Web servers with one or more Oracle WebLogic Server clusters to provide dynamic HTTP and clustered objects.
- You can use familiar firewall policies.

Using a Web server proxy at the front end of your Web application enables you to use familiar firewall policies to define your DMZ. In general, you can continue placing the Web servers in your DMZ while disallowing direct connections to the remaining Oracle WebLogic Server clusters in the architecture. The diagram in the slide depicts this DMZ policy.

However, there are some disadvantages:

- Additional administration
- Limited load balancing options

Proxy Web Server Plug-In Versus Load Balancer

- There are many advantages to using a physical load balancer instead of the proxy plug-in:
 - There is no need to configure the client plug-ins.
 - It eliminates the proxy layer, thereby reducing the number of connections.
 - There are more sophisticated load-balancing algorithms.
- There are a number of disadvantages as well:
 - Additional administration
 - Explicit configuration of “sticky” sessions for stateful Web applications



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

Proxy Web Server Plug-In Versus Load Balancer

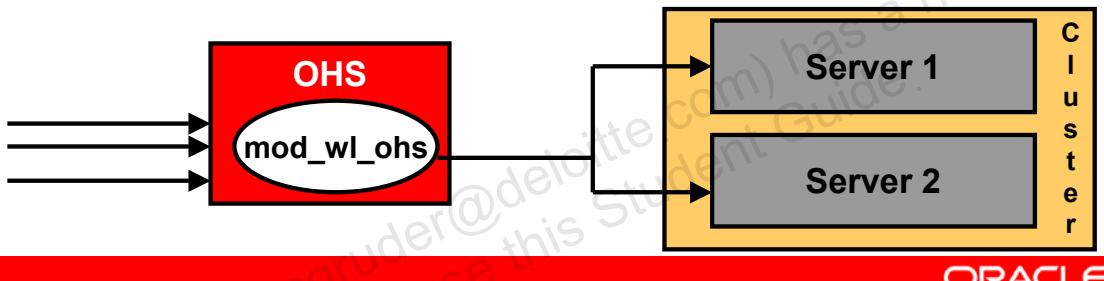
Using a load balancer directly with an Oracle WebLogic Server cluster provides several benefits over proxying servlet requests. First, using Oracle WebLogic Server with a load balancer requires no additional administration for client setup—you do not need to set up and maintain a separate layer of HTTP servers and you do not need to install and configure one or more proxy plug-ins. Removing the Web proxy layer also reduces the number of network connections that are required to access the cluster.

Using a load-balancing hardware provides more flexibility for defining the load-balancing algorithms that suit the capabilities of your system. You can use any load-balancing strategy (for example, load-based policies) that your load-balancing hardware supports. With proxy plug-ins, you are limited to a simple round-robin algorithm for clustered servlet requests.

Note, however, that using a third-party load balancer may require additional configuration if you use in-memory session state replication. In this case, you must ensure that the load balancer maintains a “sticky” connection between the client and its point-of-contact server, so that the client accesses the primary session state information. When using proxy plug-ins, no special configuration is necessary because the proxy automatically maintains a sticky connection.

Proxy Plug-Ins

- Proxy plug-ins:
 - Delegate dynamic content requests to WLS servers and balance load across a cluster in a round-robin fashion
 - Route HTTP requests to back-end WLS instances based on session cookie or URL rewriting
 - Avoid routing to failed servers in the cluster
- Oracle HTTP Server contains `mod_wl_ohs`, which is a plug-in for WLS by default.
- WLS provides plug-ins to other major Web servers as well.



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

Proxy Plug-Ins

A proxy plug-in may be essential in an environment where Oracle HTTP Server (OHS) or other Web servers serve static pages, and an Oracle WebLogic Server (possibly on a different host) is delegated to serve dynamic pages (such as JSPs or pages generated by HTTP servlets). To the end user (the browser), the HTTP responses still appear to come from the same source—the Web server running the plug-in. Oracle WebLogic Server on the back end is invisible. The HTTP-tunneling facility of the WebLogic client/server protocol can operate through the plug-in, providing access to all Oracle WebLogic Server services (not just dynamic pages).

Oracle WebLogic Server plug-ins provide efficient performance by reusing connections from the plug-in to Oracle WebLogic Server. The plug-in maintains “keep-alive” connections between the plug-in and Oracle WebLogic Server.

For documentation on plug-ins, see *Oracle Fusion Middleware Using Web Server Plug-Ins with Oracle WebLogic Server 11g Release 1 (10.3.3)*.

OHS as Proxy Web Server

OHS is a Web server that:

- Is based on Apache
- Serves static and dynamic content
- Supports content generation in many languages, such as Java, C, C++, PHP, PERL, or PL/SQL
- Contains a WebLogic Server plug-in (`mod_wl_ohs`) by default
- Can be easily integrated with other Oracle Fusion Middleware components
- Can be managed using the Fusion Middleware Control along with other components



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

OHS as Proxy Web Server

Oracle HTTP Server is based on the Apache Web server. It serves both static and dynamic content and supports applications developed in Java, PL/SQL, C, C++, PHP, or PERL. OHS supports single sign-on, clustered deployment and high availability, and Web Cache. In addition, plug-ins that are available as separate components enable integration with non-Oracle HTTP Servers.

A `mod_wl_ohs` module is available in OHS and enables you to integrate your WebLogic Server environment with OHS immediately after the configuration of the OHS instance and the domains.

Request Flow When Using OHS

- The client sends an HTTP request to OHS for access to a Java EE application.
- The `mod_wl_ohs` plug-in at OHS receives the request and determines from the cookie (in request) which WLS server should serve the request.
- If no cookie exists, the request is assigned to the next available WLS server in the cluster (round-robin algorithm).
- The WLS server that responds places the appropriate cookie in the response.
- OHS routes the response to the client (with the cookie).

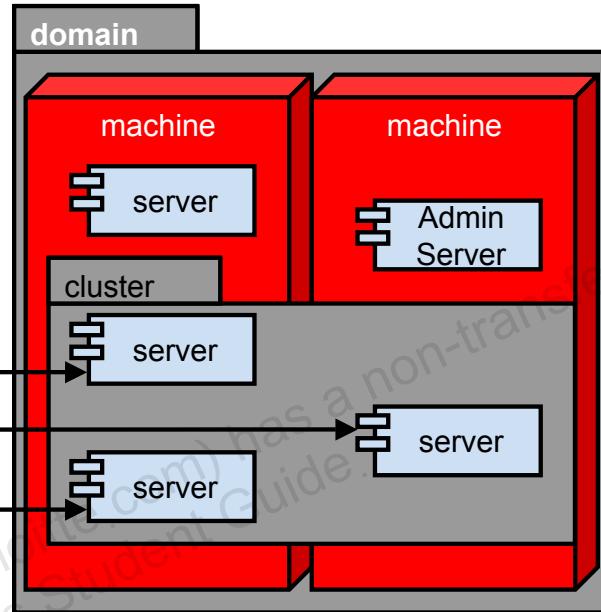
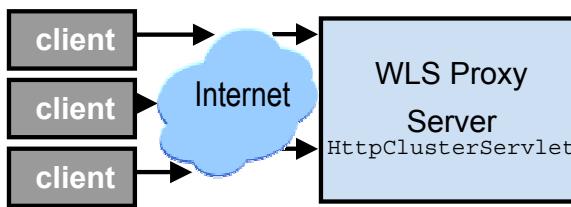
ORACLE

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

WLS HttpClusterServlet

HttpClusterServlet:

- Is deployed in the default Web application of the proxy server
- Delivers client requests in round-robin style to servers in the cluster



ORACLE

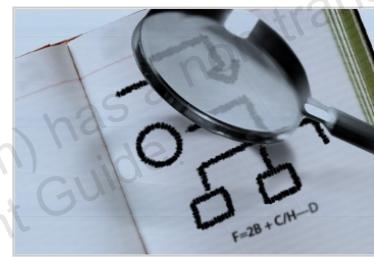
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

WLS HttpClusterServlet

HttpClusterServlet proxies the requests from an Oracle WebLogic Server to other Oracle WebLogic Server instances within a cluster. HttpClusterServlet provides load balancing and failover for the proxied HTTP requests.

Road Map

- Oracle WebLogic cluster introduction
- Cluster architecture
- Cluster communication
 - Server communication in a cluster
 - Detecting a failure
 - Multitier communication



ORACLE

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

Server Communication in a Cluster

- WebLogic Server instances in a cluster communicate with one another using:
 - IP sockets, which are the conduits for peer-to-peer communication between clustered server instances
 - IP unicast or multicast, which server instances use to broadcast availability of services and heartbeats that indicate continued availability
- Multicast broadcasts one-to-many communications among clustered instances.
- Unicast is an alternative to multicast to handle cluster messaging and communications. The unicast configuration is much easier because it does not require cross-network configuration that multicast requires.



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

Server Communication in a Cluster

Peer-to-peer communications between server instances in a cluster use IP sockets. IP sockets provide a simple, high-performance mechanism for transferring messages and data between two applications.

WebLogic Server uses IP multicast for all one-to-many communications among server instances in a cluster. This communication includes:

- **Clusterwide JNDI updates:** Each WebLogic Server instance in a cluster uses multicast to announce the availability of clustered objects that are deployed or removed locally. Each server instance in the cluster monitors these announcements and updates its local JNDI tree to reflect current deployments of clustered objects. For more details, see the section titled “Clusterwide JNDI Naming Service” later in this lesson.
- **Cluster heartbeats:** Each WebLogic Server instance in a cluster uses multicast to broadcast regular “heartbeat” messages that advertise its availability. By monitoring heartbeat messages, server instances in a cluster determine when a server instance has failed. (Clustered server instances also monitor IP sockets as a more immediate method of determining when a server instance has failed.)

IP multicast is a broadcast technology that enables multiple applications to subscribe to an IP address and port number and listen for messages. A multicast address is an IP address in the range 224.0.0.0–239.255.255.255.

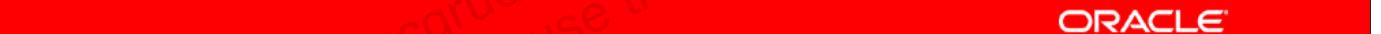
Server Communication in a Cluster (continued)

WebLogic Server provides an alternative to using multicast to handle cluster messaging and communications. Unicast configuration is much easier because it does not require cross-network configuration that multicast requires. Additionally, it reduces potential network errors that can occur from multicast address conflicts.

When creating a new cluster, it is recommended that you use unicast for messaging within a cluster.

One-To-Many Communications

- Oracle WebLogic Server uses one-to-many communication for:
 - Clusterwide JNDI updates
 - Cluster “heartbeats”
- Because all one-to-many communications occur over IP multicast, when you design a cluster, consider the following factors:
 - If your cluster spans multiple subnets, your network must be configured to reliably transmit messages.
 - A firewall can break IP multicast transmissions.
 - The multicast address should not be shared with other applications.
 - Multicast storms may occur.

The red bar contains the word "ORACLE" in white capital letters.

ORACLE

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

One-To-Many Communications

Oracle WebLogic Server uses multicast to broadcast regular “heartbeat” messages that advertise the availability of individual server instances in a cluster. The servers in a cluster listen to heartbeat messages to determine when a server has failed. (Clustered servers also monitor IP sockets as a more immediate method of determining when a server has failed.)

All servers use multicast to announce the availability of clustered objects that are deployed or removed locally. Servers monitor the announcements so that they can update their local JNDI tree to indicate the current deployments of clustered objects.

Because multicast controls the critical functions related to detecting failures and maintaining the clusterwide JNDI tree, it is important that neither the cluster architecture nor the network topology interfere with multicast communications.

If server instances in a cluster do not process incoming messages on a timely basis, increased network traffic and heartbeat retransmissions can result. The repeated transmission of multicast packets on a network is referred to as a multicast storm, and can stress the network and attached stations, potentially causing end-stations to hang or fail. Increasing the size of the multicast buffers can improve the rate at which announcements are transmitted and received, and prevent multicast storms.

One-To-Many Communications (continued)

Therefore, you should keep the following in mind.

- You may want to distribute a WebLogic Server cluster across multiple subnets in a Wide Area Network (WAN) to increase redundancy, or to distribute clustered server instances over a larger geographical area.
- Firewalls can break multicast transmissions. Although it might be possible to tunnel multicast transmissions through a firewall, this practice is not recommended for Oracle WebLogic Server clusters. Each Oracle WebLogic Server cluster should be treated as a logical unit that provides one or more distinct services to the client. Such a logical unit should not be split between security zones.
- Using the same multicast address in other applications will cause the server instances to process unnecessary messages.
- If the server instances do not process incoming messages in a timely manner, repeated transmissions on a network can cause a multicast storm.

Considerations When Using Unicast

- Unicast messaging type:
 - Is much easier to configure because it does not require cross-network configuration that multicast requires
 - Reduces potential network errors that can occur from multicast address conflicts
- You cannot mix and match cluster messaging types within a cluster.

ORACLE

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

Considerations When Using Unicast

The following considerations apply when using unicast to handle cluster communications:

- All members of a cluster must use the same message type. Mixing between multicast and unicast messaging is not allowed.
- You must use multicast if you need to support a previous version of WebLogic Server within your cluster.
- Individual cluster members cannot override the cluster messaging type.
- The entire cluster must be shut down and restarted to change the messaging type.
- JMS topics configured for multicasting can access WebLogic clusters configured for unicast because a JMS topic publishes messages on its own multicast address that is independent of the cluster address. However, the following considerations apply:
 - The router hardware configurations that allow unicast clusters may not allow JMS multicast subscribers to work.
 - JMS multicast subscribers need to be in a network hardware configuration that allows multicast accessibility.
 - For more details, see “Create and Configure Clusters” in *Programming WebLogic JMS*.

Peer-To-Peer Communications

Oracle WebLogic Server uses peer-to-peer communications for:

- Accessing nonclustered or pinned objects that reside on a remote server instance in the cluster
- Replicating HTTP session states and stateful session EJB states between a primary and a secondary server
- Accessing the clustered objects that reside on a remote server instance (typically, in a multitier cluster architecture)



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

Peer-To-Peer Communications

Proper socket configuration is crucial to the performance of an Oracle WebLogic Server cluster. Two factors determine the efficiency of socket communications in Oracle WebLogic Server:

- Whether the server's host system uses a native or a pure-Java socket reader implementation
- For systems that use pure-Java socket readers, whether or not the server is configured to use enough socket reader threads

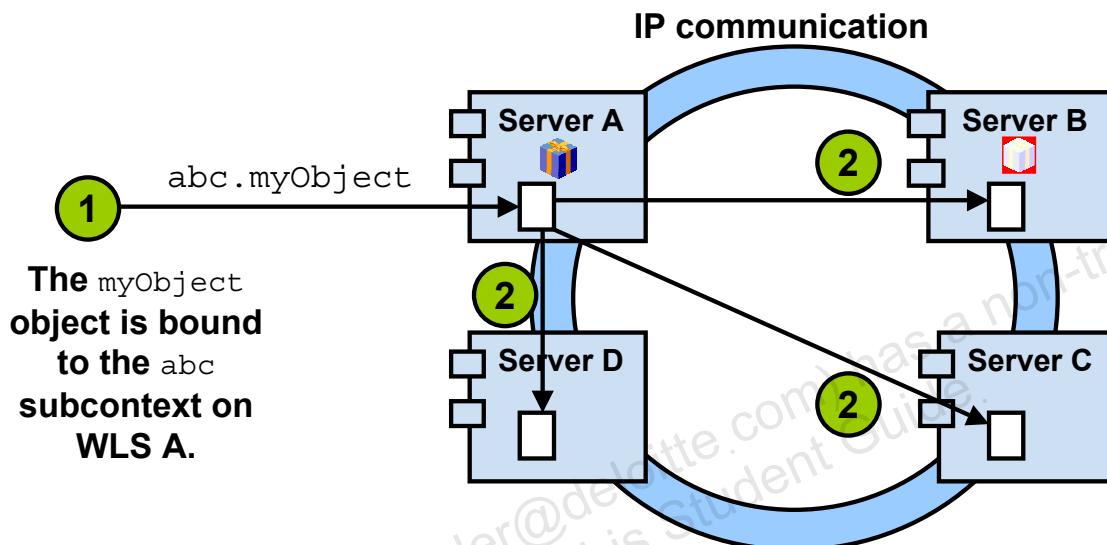
IP sockets provide a simple, high-performance mechanism for transferring messages and data between two applications. Clustered Oracle WebLogic Server instances use IP sockets for the following:

- Accessing nonclustered objects that are deployed to another clustered server instance on a different machine
- Replicating HTTP session states and stateful session EJB states between a primary and secondary server instance
- Accessing clustered objects that reside on a remote server instance (This generally occurs only in a multitier cluster architecture.)

Note: The use of IP sockets in Oracle WebLogic Server actually extends beyond the cluster scenario—all RMI communication takes place using sockets (for example, when a remote Java application accesses a remote object).

Clusterwide JNDI Naming Service

Each WebLogic Server in a cluster builds and maintains its own local copy of the clusterwide JNDI tree, which lists the services offered by all members of the cluster.



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

ORACLE

Clusterwide JNDI Naming Service

Clients access objects and services by using a JNDI-compliant naming service. Server instances in a cluster use a clusterwide JNDI tree. A clusterwide JNDI tree contains a list of locally available services and the services offered by clustered objects from other servers in the cluster. Each WebLogic Server in a cluster builds and maintains its own local copy of the clusterwide JNDI tree. As a server instance boots or as new services are dynamically deployed to a running server instance, the server instance first binds the implementations of those services to the local JNDI tree. The slide shows the following steps of clusterwide JNDI tree formation:

1. Server A has successfully bound an implementation of a clustered object into its local JNDI tree. Because the object is clustered, it offers this service to all other members of the cluster.
2. A copy of `myObject` is sent to all other WebLogic Server instances in the cluster and bound in their `abc` subcontext notifying them that this service is available on Server A.

Name Conflicts and Resolution

- Cluster-level JNDI conflicts may occur when new services are added to the cluster.
- In case of name conflicts, local binding may succeed, but the binding of other object names from other servers will fail.
- To avoid cluster-level JNDI conflicts, you must deploy all replica-aware objects to all WebLogic Server instances in a cluster.



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

Name Conflicts and Resolution

Cluster-level JNDI conflicts may occur when new services are advertised over multicast or unicast. For example, if you deploy a pinned RMI object on one server instance in the cluster, you cannot deploy a replica-aware version of the same object on another server instance.

If two server instances in a cluster attempt to bind objects using the same name, local binding may succeed. However, the server instances with conflicting names will refuse to bind the server instances' replica-aware stub in to the JNDI tree. A conflict of this type would remain until one of the two server instances was shut down or until the clustered object is undeployed from all servers.

To avoid name conflicts, deploy all cluster-level objects to all members of the cluster. Also, avoid deploying clustered and nonclustered objects in a server.

Quiz

Which is a benefit of multitier cluster architecture?

- a. Requires fewer servers compared to the basic architecture
- b. Possibility to load-balance method calls to clustered EJBs
- c. Easier security implementation
- d. None



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

Quiz

In a multitier cluster architecture where you want to load-balance EJB objects, you configure them:

- a. Within one cluster
- b. In different clusters
- c. Along with the Web-tier clients in the same server
- d. In different domains



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

Summary

In this lesson, you should have learned about:

- The benefits of the Oracle WebLogic cluster
- Basic cluster architecture
- Multitier cluster architecture
- Communication among clustered server instances
- The key criteria for selecting a suitable cluster architecture



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

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

Carl McGruder (camcgruder@deloitte.com) has a non-transferable
license to use this Student Guide.

17

Configuring a Cluster

ORACLE

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

Objectives

After completing this lesson, you should be able to:

- Prepare your environment for a cluster
- Create and configure a cluster
- Add servers to a cluster
- Start up and shut down clustered servers



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

Objectives

Scenario

The Medical Records department has decided to implement and evaluate clustering on a test application to better understand the clustering functionality. Before implementing a cluster, you need to configure the Oracle HTTP Server as the Web tier front end for your applications. You create a basic cluster using MedRecSvr2 and MedRecSvr3 managed servers. Later, you deploy and configure the test application so that HTTP session replication is enabled.

Road Map

- Preparing for a cluster
 - Cluster architecture
 - Network and security topology
 - Machines
 - Names and addresses
- Configuring a cluster



ORACLE

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

Preparing Your Environment

Before you configure a cluster, you need to prepare your environment.

- Determine your cluster architecture.
- Understand your network and security topologies.
- Choose the machines for the cluster installation.
- Identify IP addresses or DNS names, and port numbers for the server instances in the cluster.
- For proxy architectures, you can have:
 - A single firewall between untrusted clients and the Web server layer
 - A firewall between the proxy layer and the cluster
- Configure the Node Manager.



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

Preparing Your Environment

The architecture that you choose affects how you set up your cluster of servers. The cluster architecture may also require that you install or configure other resources, such as load balancers, HTTP servers, and proxy plug-ins.

Depending on the network topology that you choose, your security requirements will change.

- Some network topologies can interfere with multicast communications.
- Avoid deploying server instances in a cluster across a firewall.

A single firewall between untrusted clients and the Web server layer can be used with both the basic cluster architecture and the multitier cluster architecture. This creates a demilitarized zone around the Web servers.

A firewall between the proxy layer and the cluster means that you need to bind the clustered server instances to publicly listed DNS names. If the internal and external DNS names are not identical, you need to configure the `ExternalDNSName` property for each server instance.

The Node Manager is useful for starting a managed server that resides on a different machine than its administration server. The Node Manager also provides features that help increase the availability of managed servers in your cluster.

Hardware

- You can set up a cluster on a single computer for demonstration or development.
 - This is not practical for production environments.
- Each computer involved in a cluster should have a static IP address.
- There is no built-in limit for the number of server instances in a cluster.
 - Large multiprocessor servers can host clusters with numerous servers.
 - The recommendation is one server instance for every two CPUs.

ORACLE

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

Hardware

The main benefits of a cluster are load balancing and failover. If multiple servers in a cluster are on the same computer, these benefits are minimized. If the computer fails, all the servers on it fail and, although you may be load balancing, it is still only the computer that handles the processing.

Load balancers and proxy servers need to know which servers are in a cluster. So, in general, you need to configure the IP address of each server in a cluster in the load balancer or proxy server. If the servers are assigned to a machine with a dynamically assigned IP address, the IP address can change, and the load balancer or proxy server would not be able to find it. So ensure that you configure the cluster on machines that have static IP addresses.

IP Addresses and Host Names

- The IP address and host name information are needed for configuring and managing:
 - The administration server
 - Managed servers
 - Multicast communication
- For a production environment, use the host name resolved at DNS rather than IP addresses.
 - Firewalls can cause IP address translation errors.
- Each server should have a unique name.
- The multicast address should not be used for anything other than cluster communications.

ORACLE

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

IP Addresses and Host Names

If the internal and external DNS names of an Oracle WebLogic Server instance are not identical, use the `ExternalDNSName` attribute for the server instance to define the server's external DNS name. Outside the firewall, `ExternalDNSName` should translate to the external IP address of the server.

If clients access Oracle WebLogic Server over the default channel and T3, do not set the `ExternalDNSName` attribute—even if the internal and external DNS names of an Oracle WebLogic Server instance are not identical—to avoid unnecessary DNS lookups.

Cluster Address

- The cluster address is used to communicate with entity and session beans by constructing the host name portion of the request URLs.
- You can explicitly define the address of a cluster.
 - The cluster address should be a DNS name that maps to the IP addresses or DNS names of each Oracle WebLogic Server instance in the cluster.
- You can also have Oracle WebLogic Server dynamically generate an address for each new request.
 - Minimizes configuration
 - Ensures an accurate cluster address
- The dynamic cluster address is created in the form of:
`listenaddress1:listenport1,listenaddress2:listenport2,listenaddress3:listenport3`



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

Cluster Address

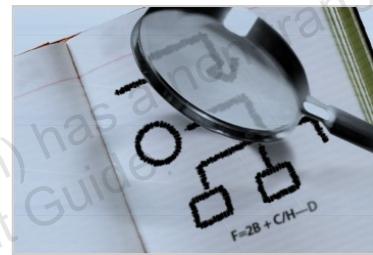
Each ListenAddress:ListenPort combination in the cluster address corresponds to the managed server and network channel that received the request. The order in which the ListenAddress:ListenPort combinations appear in the cluster address is random; the order varies from request to request.

The cluster address forms a portion of the URL that a client uses to connect to the cluster. The cluster address is used for generating EJB handles and entity EJB failover addresses. (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.)

If network channels are configured, it is possible to set the cluster address on a per-channel basis.

Road Map

- Preparing for a cluster
- Configuring a cluster
 - Administration Console
 - Configuration Wizard
 - WLST
 - Ant



ORACLE

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

Methods of Configuring Clusters

There are multiple ways to create and configure an Oracle WebLogic Server cluster:

- Configuration Wizard
- Administration Console
- WebLogic Scripting Tool (WLST)
- Java Management Extensions (JMX)
- WebLogic Server API



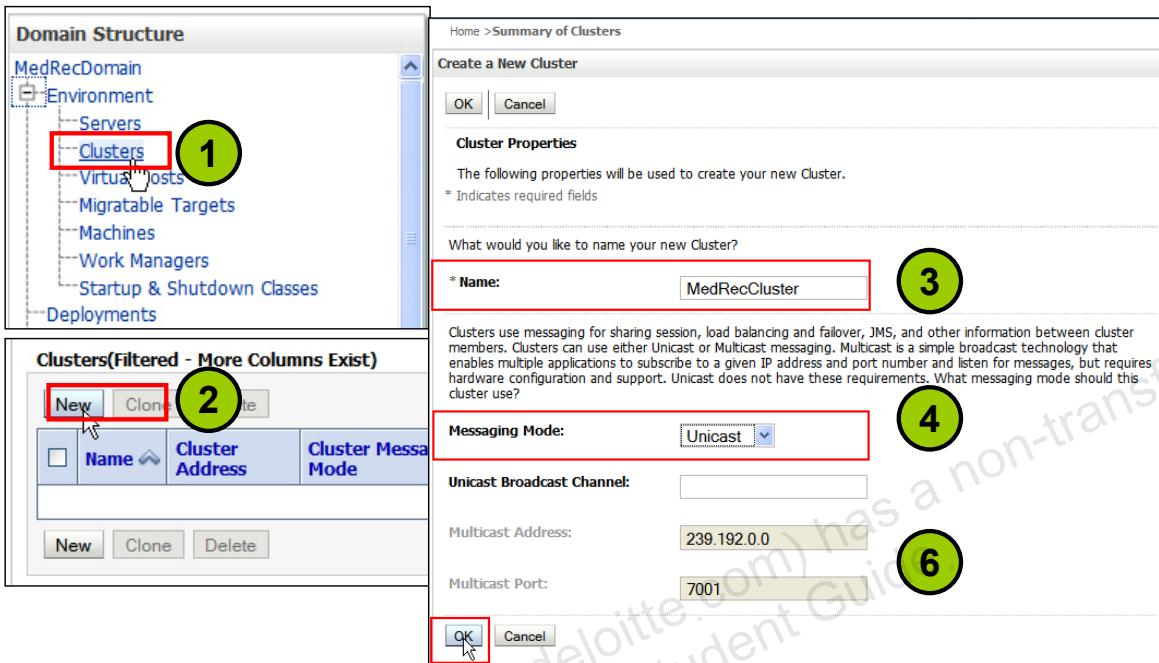
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Methods of Configuring Clusters

You can use different methods to configure a cluster.

- **Configuration Wizard:** The Configuration Wizard is the recommended tool for creating a new domain with the cluster.
- **WebLogic Server Administration Console:** If you have an operational domain within which you want to configure a cluster, you can use the Administration Console.
- **WebLogic Scripting Tool (WLST):** You can use the WLST in a command-line scripting interface to monitor and manage clusters.
- **JMX:** WebLogic Server provides a set of MBeans that you can use to configure, monitor, and manage WebLogic Server resources through JMX.
- **WebLogic Server API:** You can write a program to modify the configuration attributes, based on the configuration API provided with WebLogic Server. This method is not recommended for initial cluster implementation. For further information, refer to the documentation: *Oracle® Fusion Middleware Developing Custom Management Utilities With JMX for Oracle WebLogic Server 11g Release 1 (10.3.1)*.

Creating a Cluster by Using the Administration Console



ORACLE

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

Creating a Cluster by Using the Administration Console

To configure a cluster using the Administration Console, perform the following steps:

1. In the Administration Console, expand Environment and click Clusters.
2. Click New.
3. Enter the name of the new cluster.
4. Select the Messaging Mode that you want to use for this cluster:
 - In Oracle WebLogic Server 10.3.1 environments, unicast is the default messaging mode. Unicast requires less network configuration than multicast.
 - Multicast messaging mode is also available and may be appropriate in the environments that use previous versions of Oracle WebLogic Server. However, unicast is the preferred mode considering the simplicity of configuration and flexibility.
5. If you are using Unicast message mode, enter the Unicast Broadcast Channel. This channel is used to transmit messages within the cluster. If you anticipate high volume of traffic and your applications use session replication, you may prefer to define a separate channel for cluster messaging mode. If you do not specify a channel, the default channel is used.

Creating a Cluster by Using the Administration Console (continued)

6. If you are using Multicast message mode:
 1. Enter the multicast address of the new cluster. A multicast address is an IP address in the range from 224.0.0.0 through 239.255.255.255. The default value used by Oracle WebLogic Server is 239.192.0.0. You should avoid using x.0.0.1 multicast addresses in the range of the permitted multicast address. The multicast address you configure must be unique to a cluster and should not be shared by other clusters.
 2. Enter the Multicast Port for the new cluster. The multicast port is used by cluster members to communicate with each other. Valid values are between 1 and 65535.
7. Click OK.

Setting Cluster Attributes

The screenshot shows the Oracle WebLogic Server Administration Console interface. At the top, there is a header bar with buttons for 'New', 'Clone', and 'Delete'. Below this is a table titled 'Clusters(Filtered - More Columns Exist)' showing one cluster named 'MedRecCluster'. The table has columns for Name, Cluster Address, Cluster Messaging Mode, Migration Basis, Default Load Algorithm, Replication Type, Cluster Broadcast Channel, and Servers. The 'MedRecCluster' row shows Unicast as the messaging mode, Database as the migration basis, Round Robin as the default load algorithm, and (None) as the replication type. To the right of the table, it says 'Showing 1 to 1 of 1 Previous | Next'.

Below the table is a section titled 'Settings for MedRecCluster' with tabs for Configuration, Monitoring, Control, Deployments, Services, and Notes. The Configuration tab is selected. Under Configuration, there are tabs for General, Messaging, Servers, Replication, Migration, Singleton Services, Scheduling, Overload, Health Monitoring, and HTTP. The General tab is selected. In the General tab, the 'Name:' field is set to 'MedRecCluster'. Under 'Default Load Algorithm:', there is a dropdown menu set to 'round-robin'. Below this, there are fields for 'Cluster Address:' (empty) and 'Number Of Servers In Cluster Address:' (set to '3').

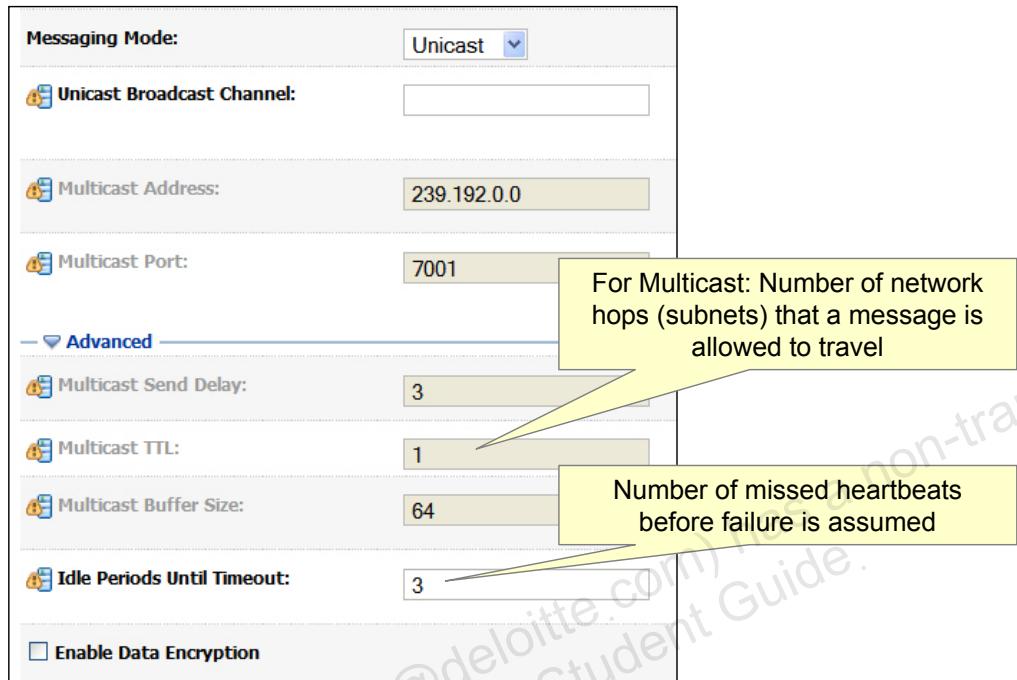
At the bottom of the screen, there is a red footer bar with the 'ORACLE' logo and the text 'Copyright © 2010, Oracle and/or its affiliates. All rights reserved.'

Setting Cluster Attributes

Some of the important cluster attributes are:

- **Default Load Algorithm:** The algorithm to be used for load balancing between replicated services if none is specified for a particular service. The *round-robin* algorithm cycles through a list of Oracle WebLogic Server instances in order. *Weight-based* load balancing improves on the round-robin algorithm by taking into account a preassigned weight for each server. In *random* load balancing, requests are routed to servers at random.
- **Cluster Address:** The address that is to be used by clients to connect to this 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.
- **Number Of Servers In Cluster Address:** The number of servers to be listed from this cluster when generating a cluster address automatically. This setting has no effect if Cluster Address is explicitly set.

Configuring Cluster Communication



ORACLE

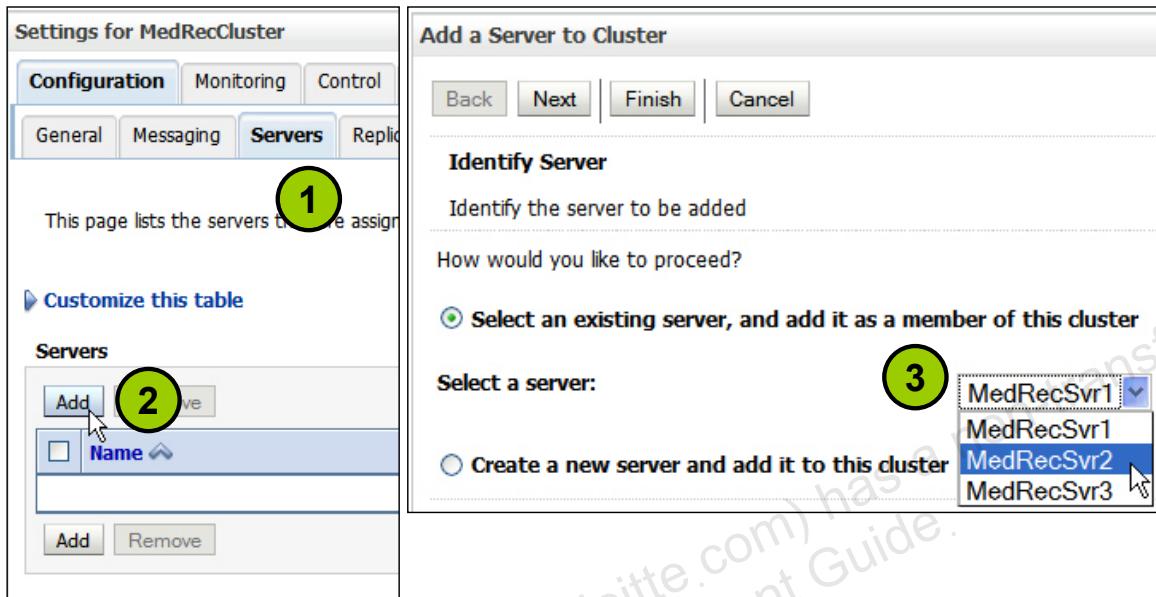
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Configuring Cluster Communication

When you configure multicast mode of communication, you may want to set up the following parameters using the Advanced configuration:

- Multicast Send Delay:** The amount of time (between 0 and 100 milliseconds) to delay sending message fragments over multicast to avoid operating system-level buffer overflow
- Multicast TTL:** The number of network hops (between 1 and 255) that a cluster multicast message is allowed to travel. 1 restricts the cluster to one subnet.
- Multicast Buffer Size:** The multicast socket send or receive buffer size (at least 64 kilobytes)
- Idle Periods Until Timeout:** The maximum number of periods that a cluster member waits before timing out a member of a cluster
- Enable Data Encryption:** The option to enable encryption of data exchanges between servers in a cluster

Adding Cluster Members: Option 1



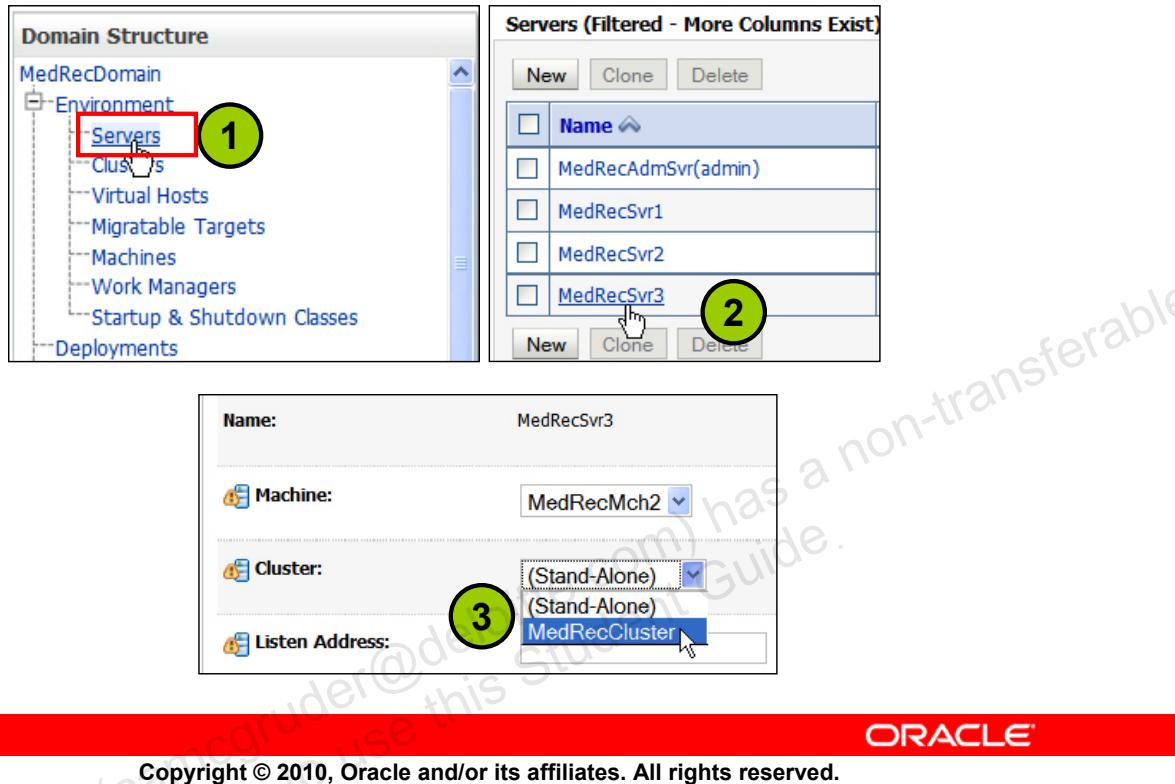
ORACLE

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

Adding Cluster Members: Option 1

1. In the Administration Console, expand Environment, and then click Clusters. Select the cluster to which you want to assign the servers. Finally, select Configuration > Servers.
2. Click Add.
3. To add an existing server to the cluster, select the “Select an existing server, and add it as a member of this cluster” option, and then select a server from the list.
To create a new server as part of a cluster, select the “Create a new server and add it to this cluster” option.

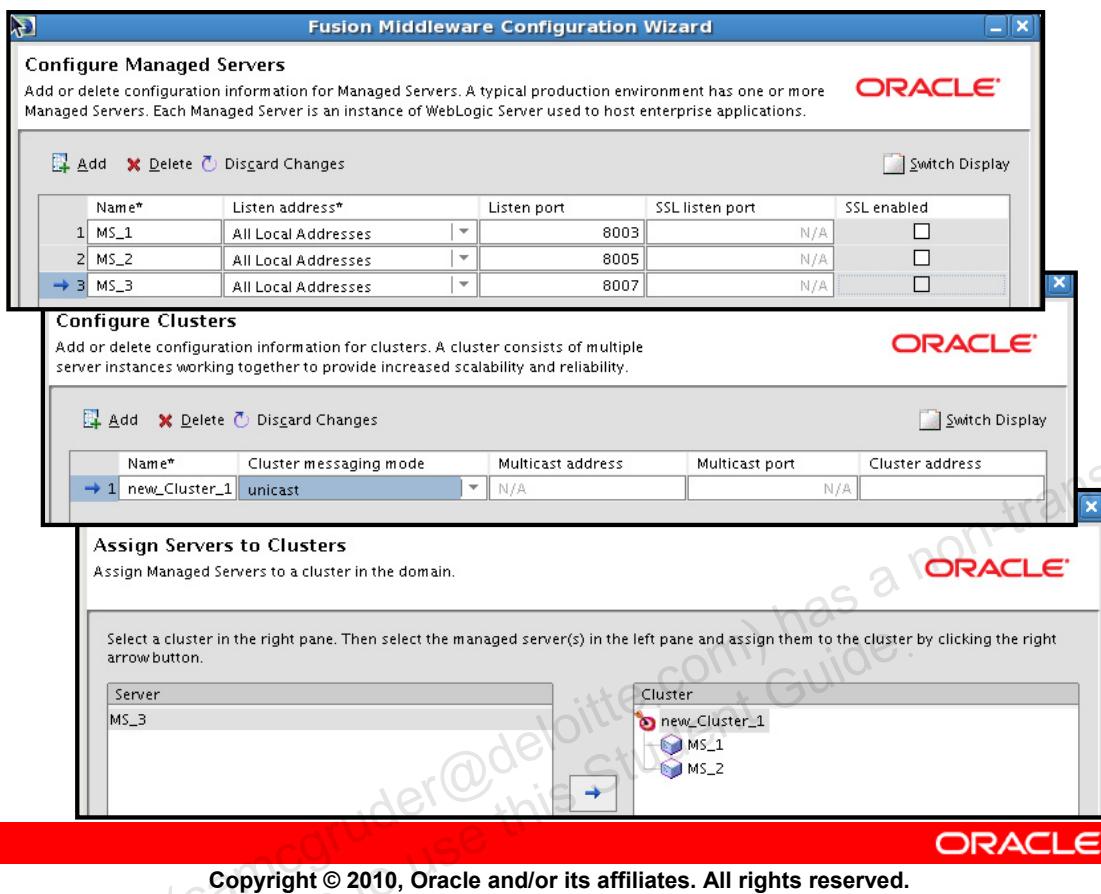
Adding Cluster Members: Option 2



Adding Cluster Members: Option 2

1. In the left pane of the Console, select Environment > Servers.
2. Select an existing server or create a new one. Confirm that the Configuration > General tab is displayed.
3. Specify whether or not this server will be a stand-alone server or will belong to a cluster.

Creating a Cluster with the Configuration Wizard



Creating a Cluster with the Configuration Wizard

You can also create and configure a cluster by using the Configuration Wizard (`./config.sh`). This is especially useful when creating a domain and you have already planned to configure clusters in the domain.

To group managed servers into clusters while creating a new domain, you can perform the following tasks in the Configuration Wizard:

- Add or delete clusters, or change the configuration of existing clusters.
- Assign the managed servers to a cluster in the domain.
- Optionally, use a managed server as an HTTP proxy for each cluster within the domain.

Creating a Cluster Using the Cluster MBean

- The Cluster MBean is used to create a cluster by using Ant or command-line tools.
- Configuring the cluster from the command line requires the combined use of Cluster and Server MBeans.
- To create new clusters within a domain, use:
 - `weblogic.management.configuration.ClusterMBean`



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

Creating a Cluster Using the Cluster MBean

It is possible to create a complete cluster from the command-line script. This is more complex than the Administration Console, but it can provide greater flexibility in making small changes to the cluster. Use the Cluster MBean to create new cluster instances. You will notice that the MBean provides all the attributes needed to configure a cluster, and the attributes and operations are provided by the Administration Console.

The configuration of clusters usually involves a coordinated effort between the Cluster MBean and Server MBeans for the servers that join and participate in a cluster. For more information about Cluster MBean, visit

http://download.oracle.com/docs/cd/E14571_01/apirefs.1111/e13951/core/index.html.

Clusters and WLST

```
connect('myuser','mypass','myhost:7001')
edit()
startEdit()
cd('/')
cmo.createCluster('HRWebCluster')
cd('/Clusters/HRWebCluster')
cluster = getMBean('/Clusters/HRWebCluster')
cd('/Servers/serverA')
cmo.setCluster(cluster)
cd('/Servers/serverB')
cmo.setCluster(cluster)
cd('/Servers/serverC')
cmo.setCluster(cluster)
activate()
disconnect()
exit()
```

Create a new cluster.

Assign cluster members.

ORACLE

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

Clusters and WLST

The example in the slide demonstrates the creation of a new cluster by using the WLST. After you create a new ClusterMBean, update each ServerMBean and assign the ClusterMBean to it.

These three lines of code:

```
cmo.createCluster('HRWebCluster')
cd('/Clusters/HRWebCluster')
cluster = getMBean('/Clusters/HRWebCluster')
```

can alternatively be written as this one line:

```
cluster = cmo.createCluster('HRWebCluster')
```

Synchronizing When Starting Servers in a Cluster

The screenshot shows two terminal windows titled "MedRecSvr2" and "MedRecSvr3". Both windows display log messages from May 20, 2010, at 6:54:00 AM UTC. The logs show the servers establishing connections with the domain-level diagnostic service, listening for announcements from the cluster, and attempting to synchronize with other members of the cluster. The logs also mention the download of the cluster JNDI tree and the start of the "async" replication service. The Oracle logo is visible in the bottom right corner of the window frame.

```

MedRecSvr2
File Edit View Terminal Tabs Help
<May 20, 2010 6:54:00 AM UTC> <Notice> <Cluster> <BEA-000197> <Listening for announcements from cluster using unicast cluster messaging>
<May 20, 2010 6:54:00 AM UTC> <Notice> <Cluster> <BEA-000197> <Waiting to synchronize with other running members of MedRecCluster.>
<May 20, 2010 6:54:00 AM UTC> <Notice> <Log Management> <BEA-170027> <The Server has established connection with the Domain level Diagnostic Service successfully.>
<May 20, 2010 6:54:00 AM UTC> <Notice> <Cluster> <BEA-000197> <Listening for announcements from cluster using unicast cluster messaging>
<May 20, 2010 6:54:00 AM UTC> <Notice> <Cluster> <BEA-000133> <Waiting to synchronize with other running members of MedRecCluster.>
<May 20, 2010 6:54:00 AM UTC> <Notice> <Cluster> <BEA-000142> <Trying to download cluster JNDI tree from server MedRecSvr2.>
<May 20, 2010 6:54:00 AM UTC> <Notice> <Cluster> <BEA-000164> <Synchronized cluster JNDI tree from server MedRecSvr2.>
<May 20, 2010 6:54:00 AM UTC> <Notice> <WebLogicServer> <BEA-000365> <Server state changed to ADMIN>
<May 20, 2010 6:54:00 AM UTC> <Notice> <WebLogicServer> <BEA-000365> <Server state changed to RESUMING>
<May 20, 2010 6:54:00 AM UTC> <Notice> <Cluster> <BEA-000162> <Starting "async" replication service with remote cluster address "null">
<May 20, 2010 6:54:00 AM UTC> <Notice> <Server> <BEA-002613> <Channel "Default[2]" is now listening on 127.0.0.1:7025 for protocols iiop, t3, CLUSTER-BROADCAST, ldap, snmp, http.>
<May 20, 2010 6:54:00 AM UTC> <Notice> <Server> <BEA-002613> <Channel "Default[1]" is now listening on fe80:0:0:0:21a:a0ff:fe0:4942c:7025 for protocols iiop, t3, CLUSTER-BROADCAST, ldap, snmp, http.>
<May 20, 2010 6:54:00 AM UTC> <Notice> <Server> <BEA-002613> <Channel "Default[3]" is now listening on 0:0:0:0:0:0:1:7025 for protocols iiop, t3, CLUSTER-BROADCAST, ldap, snmp, http.>
<May 20, 2010 6:54:00 AM UTC> <Notice> <Server> <BEA-002613> <Channel "Default" is now listening on 139.185.35.119:7025 for protocols iiop, t3, CLUSTER-BROADCAST, ldap, snmp, http.>
<May 20, 2010 6:54:00 AM UTC> <Notice> <WebLogicServer> <BEA-000330> <Started WebLogic Managed Server "MedRecSvr3" for domain "MedRecDomain" running in Production Mode>
<May 20, 2010 6:54:02 AM UTC> <Notice> <WebLogicServer> <BEA-000365> <Server state changed to RUNNING>
<May 20, 2010 6:54:02 AM UTC> <Notice> <WebLogicServer> <BEA-000360> <Server started in RUNNING mode>

MedRecSvr3
File Edit View Terminal Tabs Help
<May 20, 2010 6:56:49 AM UTC> <Notice> <Log Management> <BEA-170027> <The Server has established connection with the Domain level Diagnostic Service successfully.>
<May 20, 2010 6:56:50 AM UTC> <Notice> <Cluster> <BEA-000197> <Listening for announcements from cluster using unicast cluster messaging>
<May 20, 2010 6:56:50 AM UTC> <Notice> <Cluster> <BEA-000133> <Waiting to synchronize with other running members of MedRecCluster.>
<May 20, 2010 6:56:50 AM UTC> <Notice> <Cluster> <BEA-000142> <Trying to download cluster JNDI tree from server MedRecSvr2.>
<May 20, 2010 6:56:50 AM UTC> <Notice> <Cluster> <BEA-000164> <Synchronized cluster JNDI tree from server MedRecSvr2.>
<May 20, 2010 6:56:51 AM UTC> <Notice> <WebLogicServer> <BEA-000365> <Server state changed to ADMIN>
<May 20, 2010 6:56:51 AM UTC> <Notice> <WebLogicServer> <BEA-000365> <Server state changed to RESUMING>
<May 20, 2010 6:56:51 AM UTC> <Notice> <Cluster> <BEA-000162> <Starting "async" replication service with remote cluster address "null">
<May 20, 2010 6:56:51 AM UTC> <Notice> <Server> <BEA-002613> <Channel "Default[2]" is now listening on 127.0.0.1:7025 for protocols iiop, t3, CLUSTER-BROADCAST, ldap, snmp, http.>
<May 20, 2010 6:56:51 AM UTC> <Notice> <Server> <BEA-002613> <Channel "Default[1]" is now listening on fe80:0:0:0:21a:a0ff:fe0:4942c:7025 for protocols iiop, t3, CLUSTER-BROADCAST, ldap, snmp, http.>
<May 20, 2010 6:56:51 AM UTC> <Notice> <Server> <BEA-002613> <Channel "Default[3]" is now listening on 0:0:0:0:0:0:1:7025 for protocols iiop, t3, CLUSTER-BROADCAST, ldap, snmp, http.>
<May 20, 2010 6:56:51 AM UTC> <Notice> <Server> <BEA-002613> <Channel "Default" is now listening on 139.185.35.119:7025 for protocols iiop, t3, CLUSTER-BROADCAST, ldap, snmp, http.>
<May 20, 2010 6:56:51 AM UTC> <Notice> <WebLogicServer> <BEA-000330> <Started WebLogic Managed Server "MedRecSvr3" for domain "MedRecDomain" running in Production Mode>
<May 20, 2010 6:56:52 AM UTC> <Notice> <WebLogicServer> <BEA-000365> <Server state changed to RUNNING>
<May 20, 2010 6:56:52 AM UTC> <Notice> <WebLogicServer> <BEA-000360> <Server started in RUNNING mode>

```

ORACLE

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

Synchronizing When Starting Servers in a Cluster

To start an Oracle WebLogic Server instance that participates in a cluster, you use the same procedure as you would for starting any managed server. You identify the administration server that the instance should use. All the configuration information for the server is obtained from the configuration repository that is associated with the administration server.

If clustered server instances do not have open sockets for peer-to-peer communication, failed servers may also be detected via the Oracle WebLogic Server heartbeat. All the server instances in a cluster use multicast or unicast to broadcast regular server heartbeat messages to the other members of the cluster. Each heartbeat message contains data that uniquely identifies the server that sends the message. Servers broadcast their heartbeat messages at regular intervals of 10 seconds. In turn, each server in a cluster monitors the multicast or unicast address to ensure that the heartbeat messages of all peer servers are being sent.

If a server that is monitoring the multicast or unicast address misses three heartbeats from a peer server (that is, if it does not receive a heartbeat from the server for 30 seconds or longer), the monitoring server marks the peer server as “failed.” It then updates its local JNDI tree, if necessary, to retract the services that were hosted on the failed server. Thus, servers can detect failures even if they have no sockets open for peer-to-peer communication.

Synchronizing When Starting Servers in a Cluster (continued)

When you start a managed server in a cluster, the server instance identifies the other running server instances in the cluster by listening for heartbeats, after a warm-up period specified by the MemberWarmupTimeoutSeconds parameter in ClusterMBean. The default warm-up period is 30 seconds.

Configuring OHS as Proxy Server

- To effectively use the load balancing and failover features, you should configure a proxy.
- You can configure OHS as the proxy by:
 - Including configuration directives in `httpd.conf`
 - Creating another file with directives and setting an include directive in `httpd.conf`
- The `WebLogicCluster` directive is the most important `mod_wl_ohs` for a cluster.
- You specify the list of host names of the managed servers with their ports in the `WebLogicCluster` directive.
- If you add or remove members to or from this list, you may have to restart OHS.

The Oracle logo is displayed on a red horizontal bar.

ORACLE

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

Configuring OHS as Proxy Server

To effectively use the load balancing and failover features of the cluster, you should configure a proxy. Because OHS is already enabled with `mod_wl_ohs`, you can easily configure Oracle HTTP Server as the proxy server for the cluster. You can edit the `httpd.conf` file of the OHS instance and do one of the following:

- Set the `mod_wl_ohs` configuration directives in the `httpd.conf` file.
- Create a configuration file such as the `mod_wl_ohs.conf` file with necessary configuration directives and set an appropriate include directive in `httpd.conf`.

A typical `mod_wl_ohs.conf` file looks like this:

```
$ cat mod_wl*conf
LoadModule weblogic_module
"${ORACLE_HOME}/ohs/modules/mod_wl_ohs.so"
<IfModule mod_weblogic.c>
    WebLogicCluster wls1.com:7021,wls2.com:7021,wls3.com:7021
    ErrorPage http://myerrorpage.mydomain.com
    MatchExpression *.jsp
</IfModule>
<Location /medrec>
    SetHandler weblogic-handler
</Location>
$
```

Starting and Stopping OHS Manually

- To make configuration changes to `httpd.conf` take effect, restart OHS.
- The processing life cycle for OHS is managed by Oracle Process Manager and Notification Server (OPMN).
- The command-line interface to OPMN is `opmnctl`.
- To restart OHS, use the following command:

```
$> ./opmnctl restartproc process-type=OHS
```

- You can also stop, and then start OHS.

```
$> ./opmnctl stopproc process-type=OHS  
$> ./opmnctl startproc process-type=OHS
```

ORACLE

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

Starting and Stopping OHS Manually

Oracle HTTP Server is managed by OPMN, which manages the Oracle Application Server processes. You can use `opmnctl` to start, stop, and restart Oracle HTTP Server.

You can include the path (`<INSTANCE_HOME>/opmn/bin`) to the `opmnctl` location or change the directory to before using the `opmnctl` commands. `INSTANCE_HOME` is the location where the Web Tier instance containing this OHS instance has been configured. For example, in the classroom environment, `opmnctl` is available in `/u01/app/work/instances/bin`.

- To start the Oracle HTTP Server process in the local instance:
`$> ./opmnctl startproc process-type=OHS`
- To stop the Oracle HTTP Server process:
`$> ./opmnctl stopproc process-type=OHS`
- To determine the state of Oracle HTTP Server:
`$> ./opmnctl status`
- To restart Oracle HTTP Server:
`$./opmnctl restartproc process-type=OHS`

Verifying Access Through OHS

Get the port on which OHS is running by using:

```
$> ./opmnctl status -l

Processes in Instance: webtier
-----+-----+-----+-----+-----+-----+
ias- |process-|     |     |     |     |     |
component |type   | pid |status | uid |memused| uptime | ports
-----+-----+-----+-----+-----+-----+
ohs1   | OHS    |19582|Alive |925773433| 347092|114:46:32| https:9999,
                                                | https:4443,
                                                | http:7777
```

The actual text
formatting of the
command wraps
lines poorly.



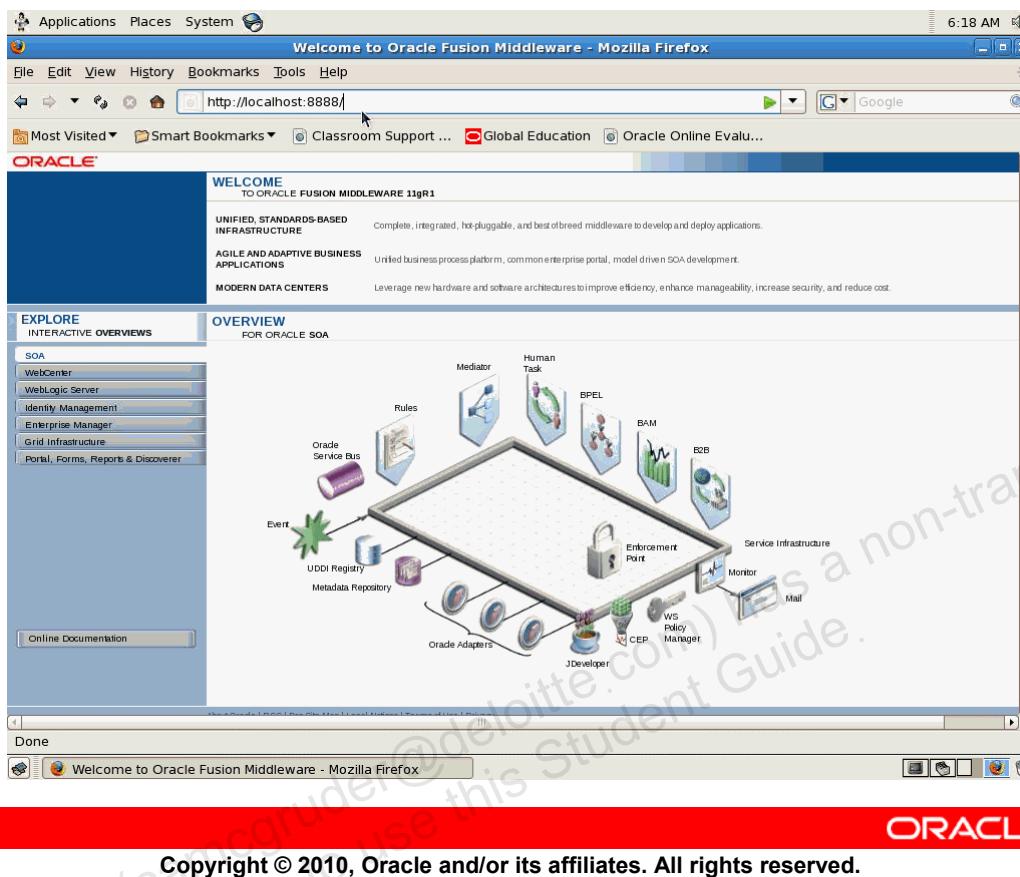
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Verifying Access Through OHS

You can verify that you are able to access the applications deployed to a cluster through OHS by directing your request to the port on which OHS is listening for requests. You can get the HTTP Listen port of OHS using the `opmnctl status -l` command. In the slide, OHS is running (HTTP) on port 7777.

Now, you can try to make a request to this port and see that the application is accessible. If it works, you will see a cool splash page as shown in the next slide.

Successful Access of OHS Splash Page



Successful Access of OHS Splash Page

Explore the clickable areas at your leisure.

Quiz

Which of the following is NOT an available configuration attribute associated with Oracle WebLogic Cluster?

- a. Messaging mode
- b. Multicast TTL
- c. Multicast port
- d. Broadcast server



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

Summary

In this lesson, you should have learned how to:

- Prepare your environment for a cluster
- Create and configure a cluster
- Add servers to a cluster
- Start up and shut down clustered servers



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

Practice 17 Overview: Configuring Clusters

This practice covers the following topics:

- Creating a cluster
- Assigning two servers to the cluster
- Verifying the port and status of Oracle HTTP Server



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

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

Carl McGruder (camcgruder@deloitte.com) has a non-transferable
license to use this Student Guide.

18

Managing Clusters

ORACLE®

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

Objectives

After completing this lesson, you should be able to:

- Deploy applications to a cluster
- Describe the replication of a session state in a cluster
- Configure replication groups
- Configure in-memory replication
- Configure Java Database Connectivity (JDBC) replication
- Configure file replication
- Configure a multitier cluster for Enterprise JavaBeans (EJB) applications



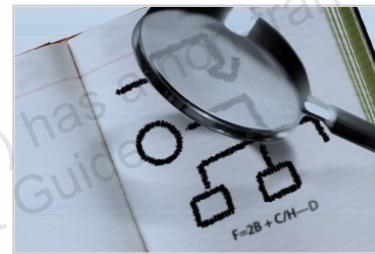
Objectives

Scenario

You deploy the application that you are using to evaluate the HTTP session failover feature. Configure Oracle HTTP Server to load balance between two managed servers in a cluster. Verify that the session failover happens appropriately.

Road Map

- Deploying applications
 - Selecting a cluster as the target
 - Two-phase deployment
 - Production redeployment
- HTTP session management
- EJB clustering

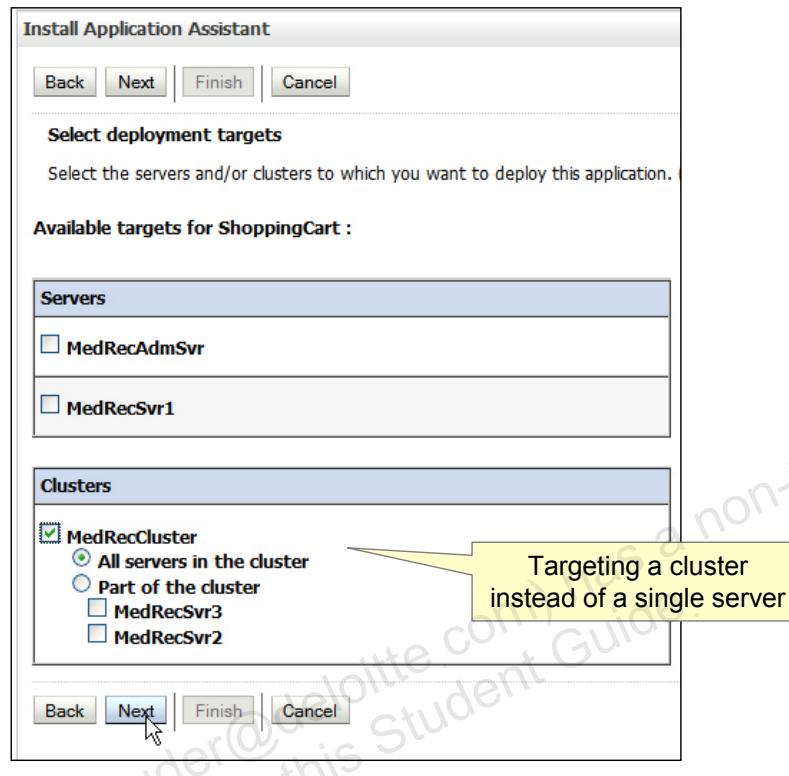


ORACLE

18 - 3

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

Deploying Applications to a Cluster



ORACLE

18 - 4

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

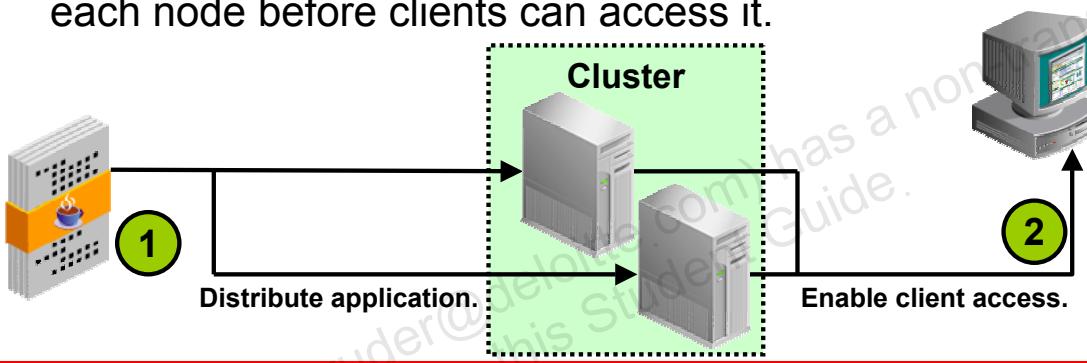
Deploying Applications to a Cluster

Regardless of the deployment tool that you use, when you initiate the deployment process, you specify the components to be deployed and the targets to which they will be deployed. The main difference between the way you deploy an application to a normal server and a cluster lies in your choice of the target. When you intend to deploy an application to the cluster, you select the target from the list of clusters and not from the list of servers.

Ideally, all servers in a cluster should be running and available during the deployment process. Deploying applications when some members of the cluster are unavailable is not recommended.

Two-Phase Deployment

- Applications are deployed using two-phase deployment (TPD).
 - Phase 1: Application components and modules are distributed to the server.
 - Phase 2: The application is deployed if phase 1 is successful and client access is permitted.
- This ensures that an application is available and active on each node before clients can access it.



18 - 5

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

ORACLE

Two-Phase Deployment

When deploying applications to a cluster, they may be packaged into a .war, .ear, or .jar file, or an exploded directory. WebLogic clusters use the concept of two-phase deployment.

- **Phase 1:** During the first phase of deployment, application components are distributed to the target server instances and the planned deployment is validated to ensure that the application components are successfully deployed. During this phase, user requests to the application being deployed are not allowed. If failures are encountered during the distribution and validation processes, the deployment is aborted on all server instances, including those on which the validation succeeded. Files that have been staged are not removed; however, container-side changes performed during the preparation are reverted.
- **Phase 2:** After the application components are distributed to targets and validated, they are fully deployed on the target server instances, and the deployed application is made available to the clients. If a failure occurs during this process, deployment to that server instance is canceled. However, a failure on one server of a cluster does not prevent successful deployment on other clustered servers.

If a cluster member fails to deploy an application, the failed application will not start in order to ensure cluster consistency. Also if the managed server is restarted with a failed application, the managed server may get started in ADMIN mode. The two-phase commit feature enables you to avoid situations in which an application is successfully deployed on one node and not on the other.

Considerations for Deploying to Cluster

- It would be good to have all the servers in the cluster running before an application is deployed to a cluster.
- If phase 2 fails on one server, the application is still deployed to other servers in the cluster.
- Do not change cluster membership while deploying applications to the cluster.
- Oracle WebLogic Server allows partial deployment of applications to a partitioned server by default.
- You can configure Oracle WebLogic Server to disallow partial deployments by using the `enforceClusterConstraints` tag.



Considerations for Deploying to Cluster

When you deploy an application to a cluster, you should run all the servers in the cluster. If a server is unavailable when the application is deployed, WebLogic switches to a relaxed deployment model. In this model, deployments continue to all other nodes. Deployment completes on the partitioned server after it becomes reachable. When the unavailable server becomes available, it may experience a performance hit as the deployment restarts on that server.

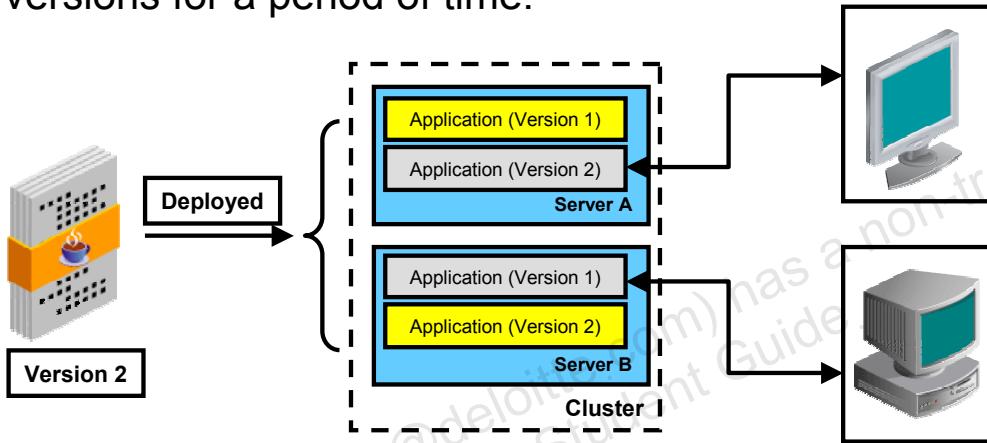
It is possible that even though a server is running, it cannot be reached by the administration server of the domain. Such an unreachable server is called a partitioned server. Oracle WebLogic Server allows deployment to such a partitioned server. This is also referred to as partial deployment. One potential problem with partial deployment is that during the synchronization with other members of the cluster—when other servers in the cluster reestablish communications with the previously partitioned server instance—the user requests to the deployed applications and the attempts to create secondary sessions on that server instance may fail causing inconsistencies in cached objects.

You can configure Oracle WebLogic Server to disallow relaxed or partial deployments by using the `enforceClusterConstraints` tag with `weblogic.Deployer`.

Production Redeployment in a Cluster

When you use production redeployment of an application in a cluster, each server instance in the cluster retires the old version when the work is complete on that server.

- Therefore, different servers may be running different versions for a period of time.



Production Redeployment in a Cluster

Production redeployment enables you to update and redeploy an application in a production environment without stopping the application or otherwise interrupting the application's availability to clients. You are saved the tasks of scheduling application down time, setting up redundant servers to host new application versions, manually managing client access to multiple application versions, and manually retiring older versions of an application.

The slide shows a cluster that contains Server A and Server B. Both servers initially run version 1 of the application. When version 2 of the application is deployed to the cluster, it is deployed to both servers in the cluster. However, because different clients are using the application on different servers, version 1 may be retired at different points. If the clients have completed using the application on Server A, any new requests are to version 2 of the application. On Server B, the client may still be interacting with version 1.

In a WebLogic Server cluster, each clustered server instance retires its local deployment of the retiring application version when the current workload is completed. This means that an application version may be retired on some clustered server instances before it is retired on other servers in the cluster. However, in a cluster failover scenario, client requests that are failed over are always handled by the same application version on the secondary server, if the application version is still available. If the same application version is not available on the secondary server, the failover does not succeed.

Road Map

- Deploying applications to clusters
- HTTP session management
 - HTTP session failover
 - In-memory replication
 - Replication groups
 - Persistence
 - JDBC-based
 - File-based
 - Best practices
- EJB clustering



ORACLE

18 - 8

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

HTTP Session Failover

- Web applications use HTTP sessions to track information in server memory for each client.
- By default, when a client fails over to another server in the cluster, its session information is lost.
- Oracle WebLogic Server supports several *Session Replication* strategies to recover sessions from failed servers:
 - In-memory replication
 - JDBC persistence
 - File persistence
- Replication is configured for each Web application within its `weblogic.xml` file.



HTTP Session Failover

Web application components, such as servlets and JavaServer Pages (JSPs), maintain data on behalf of clients using an `HttpSession` instance that is available on a per-client basis. To provide high availability of Web applications, shared access to one `HttpSession` object must be provided. `HttpSession` objects can be replicated within Oracle WebLogic Server by storing their data using in-memory replication, file system persistence, or in a database.

In a cluster, the load-balancing hardware or the proxy plug-in in Web Server redirects the client requests to any available server in the Oracle WebLogic Server cluster. The cluster member that serves the request obtains a replica of the client's HTTP session state from the available secondary server in the cluster.

HTTP Session State Replication

- Session persistence is configured using the `<session-descriptor>` element in the `weblogic.xml` deployment descriptor file.
 - Each persistence method has its own set of configurable parameters.
- You should also configure access to the cluster through a collection of Web servers with identically configured proxy plug-ins or load-balancing hardware.
- Machine definition is one of the factors that WebLogic takes into account when it chooses another server as its backup for session information.



HTTP Session State Replication

Load balancing for servlet and JSP HTTP session states can be accomplished using separate load-balancing hardware or by using the built-in load-balancing capabilities of a WebLogic proxy plug-in.

For clusters that use a bank of Web servers and WebLogic proxy plug-ins, the proxy plug-ins provide only a round-robin algorithm for distributing requests to the servlets and JSPs in a cluster.

Clusters that use a hardware load-balancing solution can use any load-balancing algorithm that the hardware supports, including advanced load-based balancing strategies that monitor the utilization of individual machines.

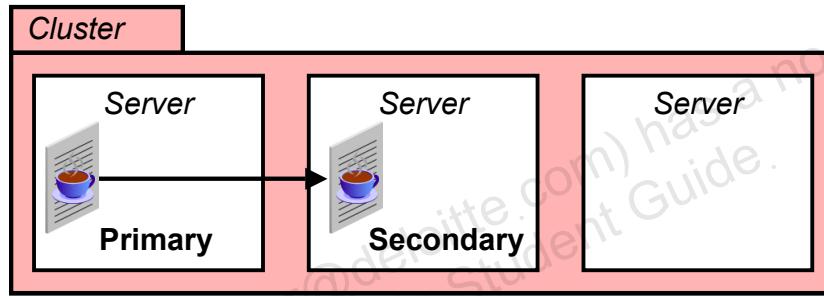
Note: This release of Oracle WebLogic Server provides Asynchronous HTTP Session Replication (AsyncRep) to improve cluster performance.

AsyncRep gives you the option to choose asynchronous session replication to the secondary server. It also provides the ability to throttle the maximum size of the queue that batches up session objects before the batched replication takes place.

AsyncRep is used to specify the asynchronous replication of data between a primary server and a secondary server. In addition, this option enables the asynchronous replication of data between a primary server and a remote secondary server located in a different cluster according to the cluster topology of MAN.

HTTP Session In-Memory Replication

- Each user's session always exists on two servers:
 - Primary
 - Secondary
- Every update to the primary session is automatically replicated on the secondary server, either synchronously (default) or asynchronously (batch).



HTTP Session In-Memory Replication

Using in-memory replication, Oracle WebLogic Server copies a session state from one server instance to another. The primary server creates a primary session state on the server to which the client first connects and a secondary replica on another Oracle WebLogic Server instance in the cluster. The replica is kept up-to-date so that it can be used if the server that hosts the Web application fails.

In-Memory Replication and Proxy Servers

- Oracle WebLogic Server uses nonpersistent cookies to track the primary and secondary servers for each client.
- Subsequent requests from the same client must be directed to the same primary server by the proxy.
- The server that is being failed over to automatically assumes the role of the primary server.



ORACLE

In-Memory Replication and Proxy Servers

To use in-memory replication for HTTP session states, you must access the Oracle WebLogic Server cluster using either a collection of Web servers with identically configured WebLogic proxy plug-ins or a load-balancing hardware.

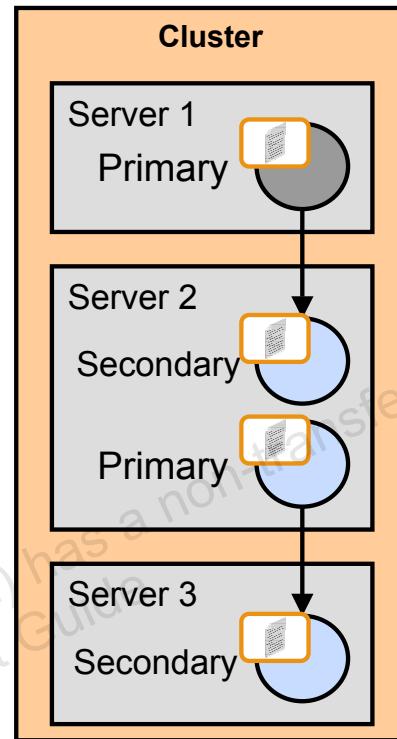
The WebLogic proxy plug-in maintains a list of Oracle WebLogic Server instances that host a clustered servlet or JSP, and forwards HTTP requests to these instances using a round-robin strategy.

Oracle WebLogic Server uses client-side cookies to keep track of the primary and secondary servers that host the client's servlet session state. If client browsers have disabled the cookie usage, Oracle WebLogic Server can also keep track of the primary and secondary servers using URL rewriting. With URL rewriting, both locations of the client session state are embedded into the URLs that are passed between the client and the proxy server. To support this feature, you must ensure that URL rewriting is enabled on the Oracle WebLogic Server cluster.

To support direct client access via the load-balancing hardware, the Oracle WebLogic Server replication system allows clients to use secondary session states regardless of the server to which the client fails over. Oracle WebLogic Server uses client-side cookies or URL rewriting to record the primary and secondary server locations. However, this information is used only as a history of the servlet session state location. When accessing a cluster via the load-balancing hardware, clients do not use the cookie information to actively locate a server after a failure.

In-Memory Replication

- WLS can replicate:
 - HttpSession objects
 - Stateful session EJBs
- Session objects exist on only two servers.
- Secondary:
 - The server is determined by the replication group and machine definition.
 - The object is created immediately after the primary object is created.
- Primary failure makes the backup object the primary object.



ORACLE

18 - 13

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

In-Memory Replication

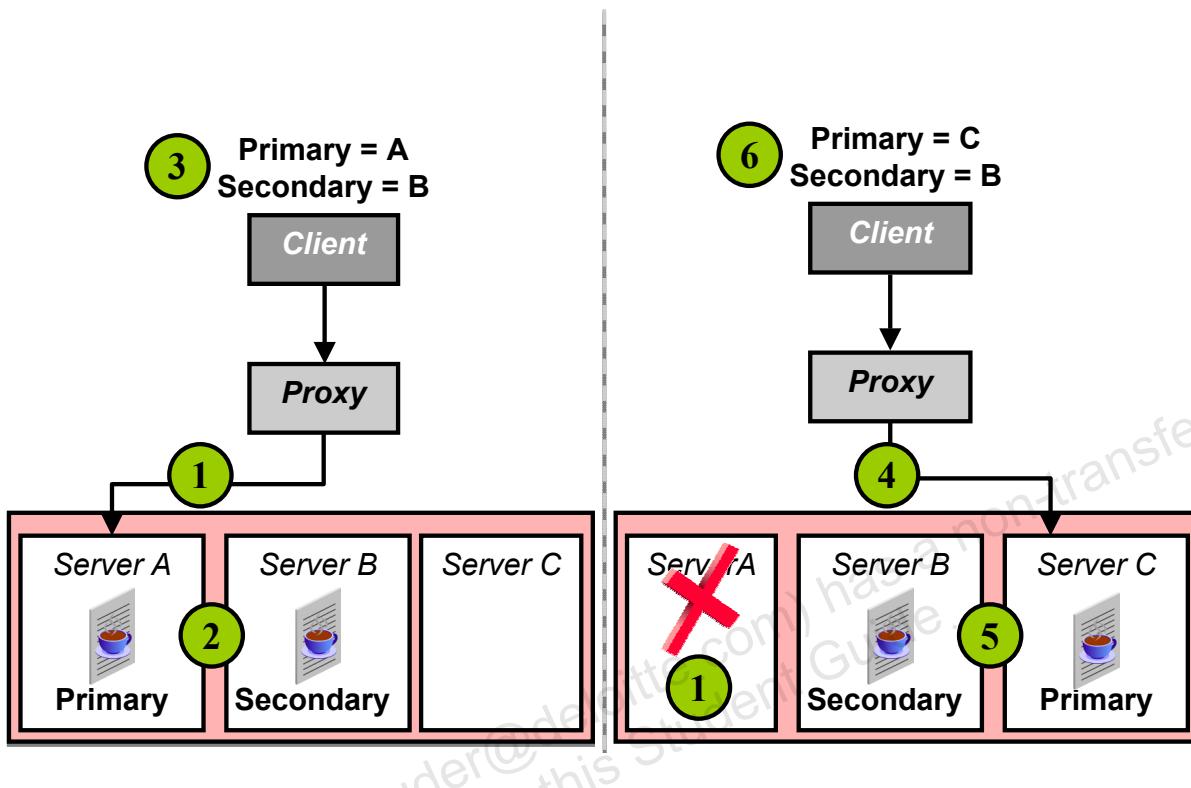
Web application components, such as servlets and JSPs, maintain data on behalf of clients using an HttpSession instance that is available on a per-client basis.

To provide high availability of Web applications, shared access to one HttpSession object must be provided. HttpSession objects can be replicated within Oracle WebLogic Server by storing their data with in-memory replication, file system persistence, or in a database.

With in-memory replication, replicated objects are not accessible on all server instances in the cluster. Rather, when an object is created, it is called the *primary object*. On another server instance, a *backup object* is created. In the event of a failure of the primary object, the backup object is promoted as the primary object. If a failover occurs, another backup object is created.

This is optimal because replication of object data must occur only between the primary and backup objects (rather than the entire cluster).

In-Memory Replication: Example



18 - 14

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

ORACLE

In-Memory Replication: Example

The graphic in the slide depicts a client accessing a Web application that is hosted in a cluster. All client requests are forwarded to the Oracle WebLogic Server cluster via a proxy, such as `HttpClusterServlet` or a Web server plug-in.

To provide failover services for the Web application, the primary server replicates the client's session state to a secondary server in the cluster. This ensures that a replica of the session state exists even if the primary server fails (for example, due to a network failure).

In the example in the slide, initially Server A is the primary server and Server B is configured as the secondary server, whereby Server A replicates the session state to Server B.

If the primary server (Server A) fails, the proxy sends the request to another member of the cluster, say Server C.

Because Server C is not secondary, it gets the session information from Server B that hosts the replica of the session state.

Now that Server C is serving the request, it becomes the primary and Server B remains secondary.

In the HTTP response, the proxy updates the client's cookie to reflect the new primary and secondary servers to account for the possibility of subsequent failovers.

Requirements for In-Memory Replication

- Subsequent requests from the same client must have access to the same primary object.
- To use in-memory replication for the HTTP session state, clients must access the cluster using one of these:
 - The load-balancing hardware (WLS aware)
 - Oracle HTTP Server with the `mod_wl_ohs` module
 - A collection of Web servers, or a single Web server, with WebLogic proxy plug-ins (configured identically)
 - Oracle WebLogic Server configured with `HttpClusterServlet`



Requirements for In-Memory Replication

Proxy Requirements

The WebLogic proxy plug-ins maintain a list of Oracle WebLogic Server instances that host a clustered servlet or JSP and forward HTTP requests to these instances by using a simple round-robin strategy.

The supported Web servers and proxy software include:

- Oracle HTTP Server with the `mod_wl_ohs` module configured
- Oracle WebLogic Server with `HttpClusterServlet`
- Netscape Enterprise Server with the Netscape (proxy) plug-in
- Apache with the Apache Server (proxy) plug-in
- Microsoft Internet Information Server with the Microsoft-IIS (proxy) plug-in

Load Balancer Requirements

If you choose to use load-balancing hardware instead of a proxy plug-in, you must use hardware that supports secure sockets layer (SSL) persistence and passive cookie persistence. Passive cookie persistence enables Oracle WebLogic Server to write cookies through the load balancer to the client. The load balancer, in turn, interprets an identifier in the client's cookie to maintain the relationship between the client and the primary Oracle WebLogic Server that hosts the HTTP session state.

Configuring In-Memory Replication

1. Configure the proxy server (if applicable).
2. Optionally, define replication groups or machines, or both.
3. Specify the persistence type in the `weblogic.xml` deployment descriptor; the options include:
 - replicated
 - replicated-if-clustered
 - async-replicated
 - async-replicated-if-clustered

```
...
<session-descriptor>
    <persistent-store-type>replicated</persistent-store-type>
</session-descriptor>
...
```



Configuring In-Memory Replication

Set the persistent store method to one of the following options:

- `memory`: Disables persistent session storage
- `replicated`: Enables replication of session data across the clustered servers, and the session data is not persistent
- `replicated_if_clustered`: Replicates the in-effect persistent-store-type if the Web application is deployed on a clustered server; otherwise, the default is `memory`
- `async-replicated`: Enables asynchronous session replication in an application or a Web application
- `async-replicated-if-clustered`: Enables asynchronous session replication in an application or Web application when deployed to a cluster environment. If deployed to a single server environment, the session persistence/replication defaults to in-memory.
This allows testing on a single server without deployment errors.
- `file`: Uses file-based persistence
- `async-jdbc`: Enables asynchronous JDBC persistence for HTTP sessions in an application or a Web application
- `j dbc`: Uses a database to store persistent sessions
- `cookie`: Stores all session data in the user's browser

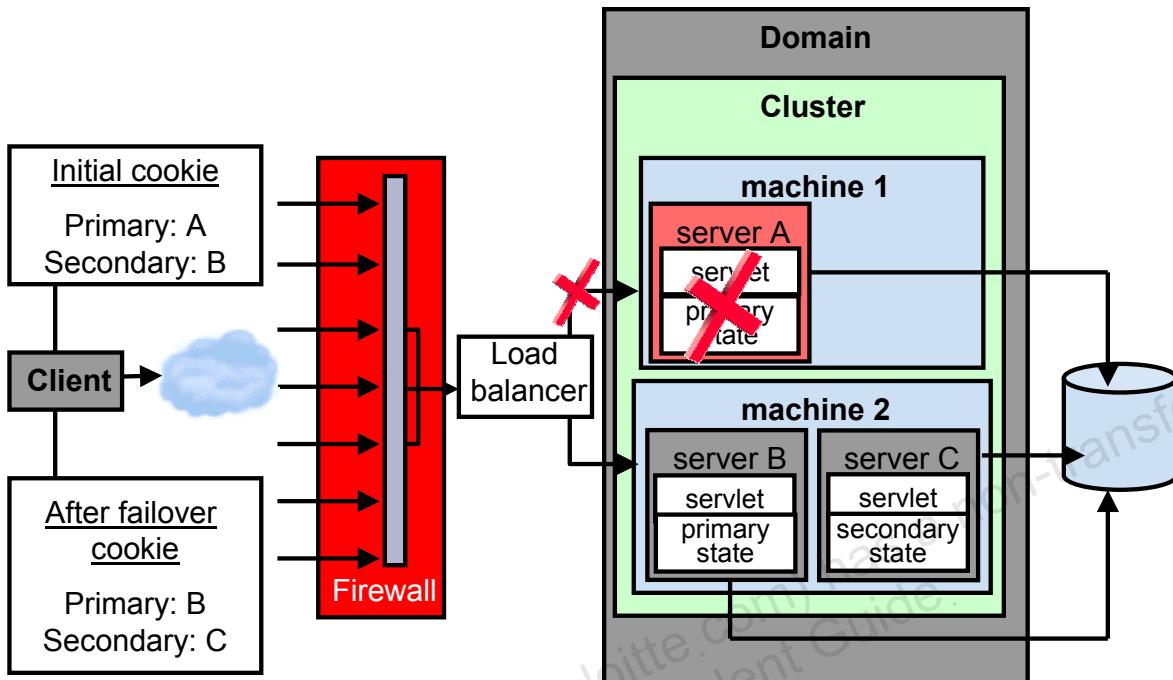
Configuring In-Memory Replication (continued)

Cookie-based session persistence provides a stateless solution for session persistence by storing all session data in a cookie in the user's browser. Cookie-based session persistence is most useful when you do not need to store large amounts of data in the session. Cookie-based session persistence can simplify management of your Oracle WebLogic Server installation because clustering failover logic is not required. Because the session is stored in the browser, and not on the server, you can start and stop Oracle WebLogic Servers without losing sessions.

Note that cookies can persist only string data and there is no security on data because cookies are passed to and from the browser in clear text.

In the `<session-param>` element of `weblogic.xml`, set the `PersistentStoreType` attribute to `cookie`. Optionally, set a name for the cookie using the `PersistentStoreCookieName` attribute. The default is `WLCOOKIE`.

Failover with Load Balancer



18 - 18

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

ORACLE

Failover with Load Balancer

When the client first makes a request, it is sent to server A. Because the application uses either an HTTP session or a stateful session bean, the client becomes pinned to a server. A cookie is written to the client machine stating that the primary state is stored on server A and the secondary on server B. Where the secondary state is stored is chosen based on machines and replication groups.

When server A fails, requests from the client go to any available server in the cluster. If it did not host the secondary, it will sync the session from the old secondary and become the new primary.

Replication Groups

- A replication group is a logical grouping of related servers in a cluster.
- WebLogic Server enables you to determine where to put in-memory backup objects using replication groups.
- WebLogic Server attempts to:
 - Send backup objects to a preferred secondary replication group, if it is configured
 - Send backup objects to a different machine
 - Avoid sending backup objects to servers in the same replication group



Replication Groups

By default, Oracle WebLogic Server attempts to create replicas of certain services on a machine other than the one that hosts the primary service.

Oracle WebLogic Server enables you to further control where the secondary states are placed by using replication groups. A replication group is a preferred list of clustered instances to use for storing session state replicas in-memory. When you configure a server instance that participates in a cluster, you can assign the server instance membership in a replication group. You can also assign a preferred secondary replication group to be considered for replicas of the primary HTTP session states that reside on the server.

When a client attaches to a cluster and creates an instance of a service, that service instance is automatically replicated in Oracle WebLogic Server (such as an HttpSession or a stateful session EJB). Oracle WebLogic Server instance that hosts the primary object honors the preferred secondary replication group if it is configured. Otherwise, a secondary on a remote machine is chosen for replication before trying to replicate to the local server.

An administrator can configure replication groups to operate such that secondary objects for replicated services always reside on different hardware. In earlier versions of Oracle WebLogic Server, the cluster would ensure that a replicated service exists on a different machine. However, because one computer can host multiple IP addresses and thus multiple machines, a replicated instance might not be protected from a general hardware failure. The creation of replication groups solves this issue.

Replication Groups

- Replication groups:
 - Represent a subset of servers within a cluster
 - Help to determine the placement of secondary sessions (for example, avoid replicating within the same room)
 - Are not explicitly defined in the console-like machines and clusters
- WLS attempts to:
 - Send secondary sessions to servers that are assigned to the *preferred secondary replication group* of the primary server
 - Avoid sending secondary sessions to servers that are assigned to the same replication group as the primary server



Replication Groups (continued)

By default, Oracle WebLogic Server attempts to create session state replicas on a machine other than the one that hosts the primary session state. You can further control where secondary states are placed using replication groups. A replication group is a preferred list of clustered servers to be used for storing session state replicas.

Using the Oracle WebLogic Server Console, you can define unique names for machines that host individual server instances. These machine names can be associated with the new Oracle WebLogic Server instances to identify where the servers reside in your system.

Machine names are used to indicate servers that run on the same machine. For example, you would assign the same machine name to all server instances that run on the same machine or the same server hardware.

If you are not running multiple Oracle WebLogic Server instances on a single machine, you need not specify the Oracle WebLogic Server machine names. Servers without a machine name are treated as though they reside on separate machines.

When you configure a clustered server instance, you can assign the server to a replication group and a preferred secondary replication group for hosting replicas of the primary HTTP session states that are created on the server.

Configuring Replication Groups

Select each server in a cluster and assign each a pair of replication groups.

Use this page to define a cluster configuration for this server. A WebLogic Server cluster is a group of servers that work together to provide a scalable and reliable application platform.

Replication Group:	RepGroupBlue	Defines preferred clustered instances considered for hosting replicas of the primary HTTP session states created on the server. More Info...
Preferred Secondary Group:	RepGroupRed	Defines secondary clustered instances considered for hosting replicas of the primary HTTP session states created on the server. More Info...
Cluster Weight:	100	The proportion of the load that this server will bear, relative to other servers in a cluster. More Info...
Interface Address:		The IP address of the NIC that this server should use for multicast traffic. More Info...

ORACLE

18 - 21

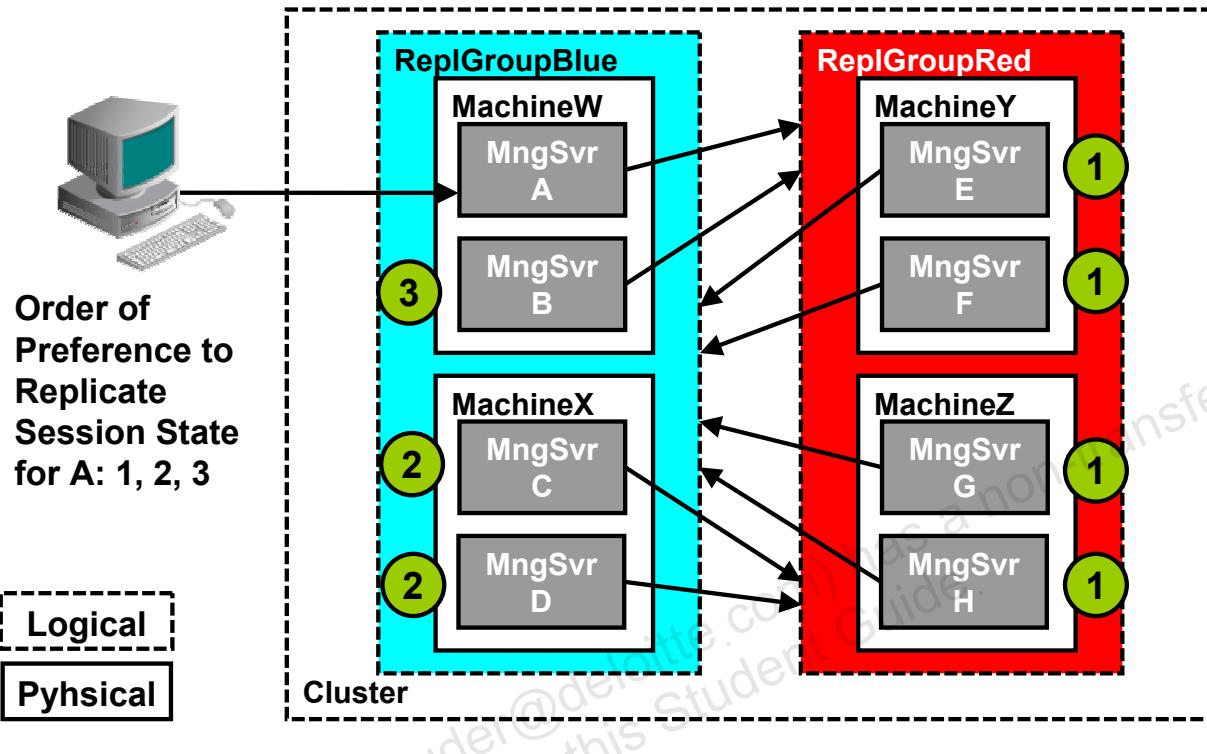
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Configuring Replication Groups

If a cluster hosts servlets or stateful session EJBs, you might want to create replication groups of the Oracle WebLogic Server instances to host the session state replicas. The server from the preferred Secondary replication group will be chosen if configured. It is the Administrator's responsibility to specify the members of replication group correctly. It is advisable not to choose a server which is co-located. If replication groups are not configured (or the preferred secondary group is down), WebLogic Server chooses a secondary which is on any remote machine. If WebLogic Server still does not have a secondary server, it will pick one from a local machine.

- Replication Group:** The name of the replication group to which the server belongs. It is recommended that you group together all servers that have a relationship with one another (for example, servers that run on the same machine). For greater flexibility, you can define a different replication group for each server.
- Preferred Secondary Group:** The name of the replication group to use to host the replicated HTTP session states for the server. You should select a secondary group in which all servers run on a different machine than the replication group's servers. For greater flexibility, you can select a secondary replication group that contains a single server running on a different machine.

Replication Groups for Different Criteria



18 - 22

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

ORACLE

Replication Groups for Different Geographic Locations

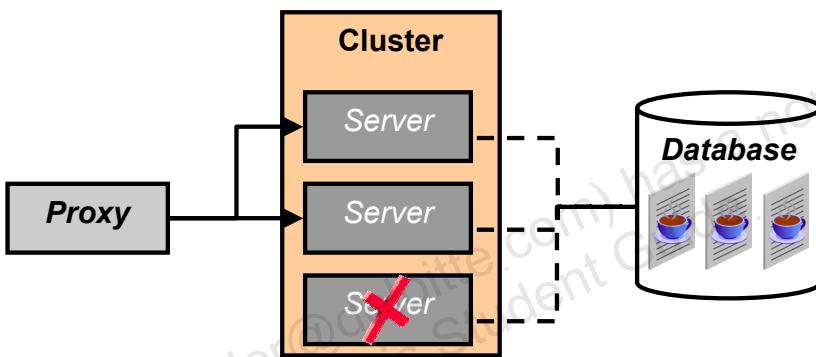
This shows an installation with eight managed servers spread out across four machines, but two machines, W and X, share one commonality, while Y and Z share another. For example: machines W and X are on the same rack, whereas Y and Z are on a different rack (or, perhaps they have different power supplies, or different locations). So, you place all servers from machines W and X in one replication group (GroupBlue) and all others in the other (GroupRed). All servers in GroupBlue choose the GroupRed as their secondary, and vice versa. (MngSvrA is in the GroupBlue but prefers to replicate to any server in the GroupRed.)

When MngSvrA gets a session, it will avoid MngSvrC and MngSvrD even though those servers are on a different machine. This is because they are on the same rack (or power supply), and if MngSvrA goes down, MngSvrC may go down as well. Because you want to mitigate that risk, servers on another rack (or power supply) from MngSvrA (for example, MngSvrE-MngSvrH) are better choices than MngSvrC or MngSvrD. Thus, replication groups allow you to account for risk factors beyond machine.

(Technically, the machines are a logical construct, but 99% of the time they align with a physical boundary.)

HTTP Session Persistence Using JDBC

- HTTP sessions can be persisted to a database using a common JDBC data source.
- The required Data Definition Language (DDL) file is available in the documentation.
- All members of the cluster have access to any client's session for failover purposes (no primary or secondary).



ORACLE

18 - 23

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

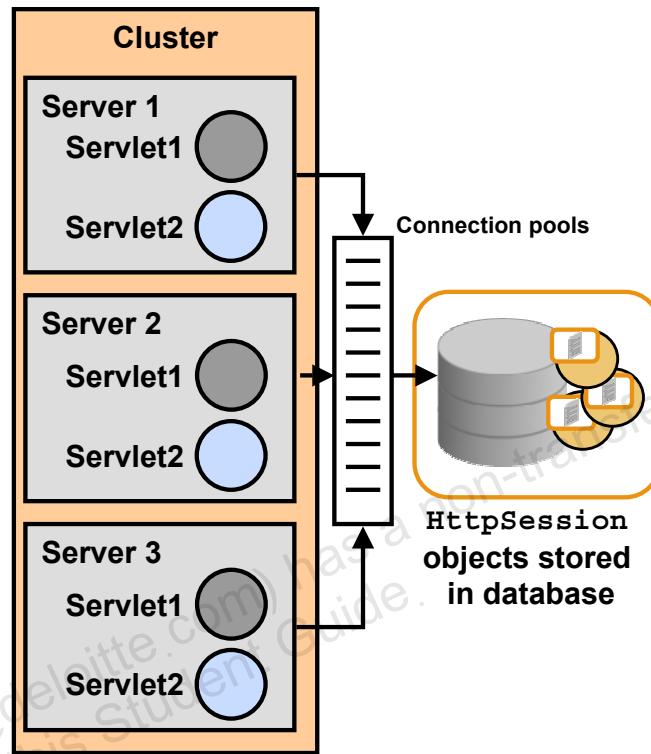
HTTP Session Persistence Using JDBC

With persistent JDBC replication, a database is configured for storing `HttpSession` objects. After the database is configured, each server instance in a cluster uses an identical connection pool to share access to the database.

Whenever a Web application creates or uses a session object, the WebLogic Web container stores the session data persistently in the database. When a subsequent client request enters the cluster, any server in the cluster can handle the request. Each server in the cluster has identical access to the persistent store where it can look up the information needed to satisfy the client's request. This technique provides good failover capability because any server in the cluster can resolve a client's request, but there is a significant performance reduction due to the many database synchronizations required in a large Web-based system.

HTTP Session Persistence Using JDBC

- All server instances have access to all sessions.
- Subsequent requests from the same client can be handled by any server.
 - Great failover capability
 - Significant performance reduction
- Changing session objects causes (slow) database synchronization.



HTTP Session Persistence Using JDBC (continued)

Whenever a servlet creates or uses a session object, the servlet stores the session data persistently in the database. When a subsequent client request enters the cluster, any server in the cluster can handle the request. Each server in the cluster has identical access to the persistent store where it can look up the information needed to satisfy the client's request. This technique provides for good failover capability because any server in the cluster can resolve a client's request, but there is significant performance reduction due to the many database synchronizations required in a large Web-based system.

Session persistence is not used for storing long-term data between sessions. That is, you should not rely on a session still being active when a client returns to a site at some later date. Instead, your application should record long-term or important information in a database.

You should not attempt to store long-term or limited-term client data in a session. Instead, your application should create and set its own cookies on the browser. Examples of this include an auto-login feature where the cookie lives for a long period or an auto-logout feature where the cookie expires after a short period of time. Here, you should not attempt to use HTTP sessions; instead, you should write your own application-specific logic.

Note that even though it is legal (according to the HTTP Servlet specification) to place any Java object in a session, only those objects that are serializable are stored persistently by Oracle WebLogic Server.

Configuring JDBC Persistence

1. Create the required table in the database.
2. Create a JDBC data source that has read/write privileges for your database.
3. Configure JDBC session persistence in the `weblogic.xml` deployment descriptor.

```
...
<session-descriptor>
  <persistent-store-type>jdbc</persistent-store-type>
  <persistent-store-pool>MyDataSource</persistent-store-pool>
</session-descriptor>
...
```



Configuring JDBC Persistence

Set up a database table named `wl_servlet_sessions` for JDBC-based persistence. The connection pool that connects to the database needs to have read/write access for this table. Create indexes on `wl_id` and `wl_context_path` if the database does not create them automatically. Some databases create indexes automatically for primary keys.

Set the `persistent-store-type` parameter in the `session-descriptor` element in the `weblogic.xml` deployment descriptor file to `j dbc`.

Set a JDBC connection pool to be used for persistence storage with the `persistent-store-pool` parameter in the `session-descriptor` element in the `weblogic.xml` deployment descriptor file. Use the name of a connection pool that is defined in the Oracle WebLogic Server Administration Console.

You can use the `j dbc-connection-timeout-secs` parameter to configure the maximum duration that the JDBC session persistence should wait for a JDBC connection from the connection pool, before failing to load the session data.

To prevent multiple database queries, Oracle WebLogic Server caches recently used sessions. Recently used sessions are not refreshed from the database for every request. The number of sessions in cache is governed by the `cache-size` parameter in the `session-descriptor` element of the Oracle WebLogic Server-specific deployment descriptor, `weblogic.xml`.

JDBC Persistent Table Configuration

A database table named WL_SERVLET_SESSIONS must exist with read/write access:

	Column Head	Column Data Type
Primary Key	WL_ID	char, 100 variable width char
	WL_CONTEXT_PATH	char, 100 variable width char
	WL_CREATE_TIME	numeric, 20 digits
	WL_IS_VALID	char, 1 character
	WL_SESSION_VALUES	BLOB, very large
	WL_ACCESS_TIME	numeric, 20 digits
	WL_IS_NEW	numeric, 20 digits
	WL_MAX_INACTIVE_INTERVAL	integer



JDBC Persistent Table Configuration

In the database that is mapped to the session persistence connection pool, you must configure a single table, WL_SERVLET_SESSIONS, which holds the values of all active session objects. The user specified with access to this table needs read/write/insert/delete access on the table to effectively manage the objects. The table requires the following eight columns:

- **WL_ID:** The session ID is used as the database primary key along with WL_CONTEXT_PATH. This is a variable-width alphanumeric data type of up to 100 characters.
- **WL_CONTEXT_PATH:** This is the context. This column is used with WL_ID as the primary key. This is a variable-width alphanumeric data type of up to 100 characters.
- **WL_IS_NEW:** This value is True as long as the session is classified in the “new” state by the Servlet engine. This is a single char column.
- **WL_CREATE_TIME:** This is the time when the session was originally created. This is a numeric column, 20 digits.
- **WL_IS_VALID:** This parameter is True when the session is available to be accessed by a servlet. It is used for concurrency purposes. This is a single char column.
- **WL_SESSION_VALUES:** This is the actual session data. It is a BLOB column.
- **WL_ACCESS_TIME:** This indicates the time when the session was last accessed. This is a numeric column, 20 digits.
- **WL_MAX_INACTIVE_INTERVAL:** This indicates the number of seconds between client requests before the session is invalidated. A negative time value indicates that the session should never time out. This is an integer column.

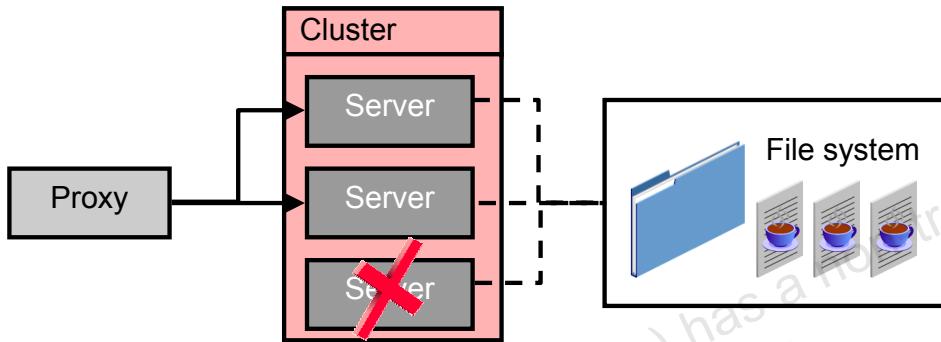
JDBC Persistent Table Configuration (continued)

The following is an example SQL statement to create this table, for Oracle Database:

```
create table wl_servlet_sessions ( wl_id VARCHAR2(100) NOT NULL,
wl_context_path VARCHAR2(100) NOT NULL, wl_is_new CHAR(1),
wl_create_time NUMBER(20), wl_is_valid CHAR(1), wl_session_values
LONG RAW, wl_access_time NUMBER(20), wl_max_inactive_interval
INTEGER, PRIMARY KEY (wl_id, wl_context_path) );
```

HTTP Session Persistence Using Files

File persistence is similar to JDBC persistence, but it persists sessions to a highly available file system.



HTTP Session Persistence Using Files

The session state may also be stored in a file. For file-based persistence:

- You must create the directory in which to store the file
- The file must have the appropriate access privileges

Configuring File Persistence

1. Create a folder shared by all servers on the cluster on a highly available file system.
2. Assign read/write privileges to the folder.
3. Configure file session persistence in the `weblogic.xml` deployment descriptor.

```
...
<session-descriptor>
    <persistent-store-type>file</persistent-store-type>
    <persistent-store-dir>/mnt/wls_share</persistent-store-dir>
</session-descriptor>
...
```



Configuring File Persistence

In the `weblogic.xml` deployment descriptor file, set the `persistent-store-type` parameter in the `session-descriptor` element to `file`.

Set the directory where Oracle WebLogic Server stores the sessions using the `persistent-store-dir` parameter. You must create this directory and make sure that appropriate access privileges are assigned to the directory.

Ensure that you have enough disk space to store the number of valid sessions multiplied by the size of each session. You can find the size of a session by looking at the files created in the location indicated by the `persistent-store-dir` parameter. Note that the size of each session can vary as the size of serialized session data changes.

Each server instance has a default persistent file store that requires no configuration. Therefore, if no directory is specified, a default store is automatically created in the `<server-name>\data\store\default` directory. However, the default store is not shareable among clustered servers.

Other options for `<persistent-store-type>`:

`memory`: When you use memory-based storage, all session information is stored in memory and is lost when you stop and restart Oracle WebLogic Server. To use memory-based, single-server, nonreplicated persistent storage, set the `PersistentStoreType` attribute in the `<session-param>` element of the `weblogic.xml` file to `memory`.

Configuring File Persistence (continued)

`cookie`: Cookie-based session persistence provides a stateless solution for session persistence by storing all session data in a cookie in the user's browser. Cookie-based session persistence is most useful when you do not need to store large amounts of data in the session.

Cookie-based session persistence can simplify management of your Oracle WebLogic Server installation because clustering failover logic is not required. Because the session is stored in the browser, and not on the server, you can start and stop Oracle WebLogic Servers without losing sessions. But remember that cookies can persist only string data and that there is no security on the data because cookies are passed to and from the browser in clear text.

To set up cookie-based session persistence:

- In the `<session-param>` element of `weblogic.xml`, set the `PersistentStoreType` attribute to `cookie`
- Optionally, set a name for the cookie using the `PersistentStoreCookieName` attribute. The default is `WLCOOKIE`.

HTTP State Management Best Practices

- Create WLS machines.
- Use replication groups.
- Choose replication strategy.
- Use the ServerDebugConfig MBean.
- Ensure that objects are serializable.



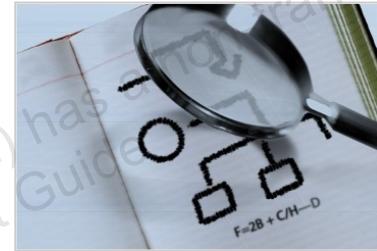
HTTP State Management Best Practices

This is a summary of the points from the previous section:

- Create WLS machines if you are replicating the state across servers on different physical machines.
- Use replication groups to define the failover strategy.
- Choose the most appropriate replication strategy depending on the application needs and architecture.
- Use the ServerDebugConfig MBean to track session replication problems.
- Ensure that objects placed in replicated sessions are serializable.

Road Map

- Deploying applications to clusters
- HTTP session management
- EJB clustering
 - EJB clustering deployment descriptors
 - Configuring stateless session beans
 - Configuring stateful session beans
 - EJB best practices



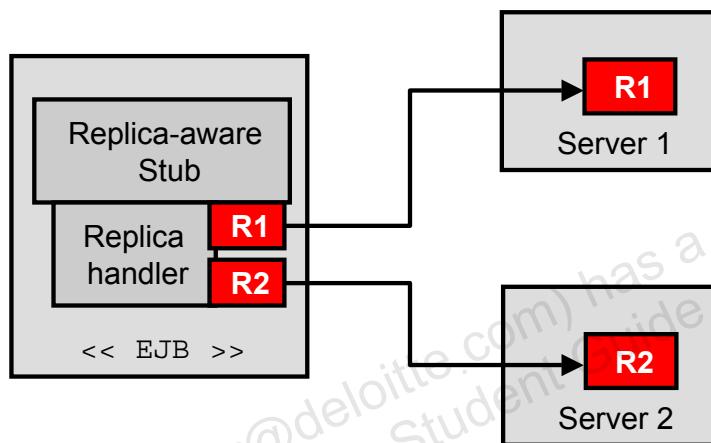
ORACLE

18 - 32

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

Clustering EJB Objects: Replica-Aware Stub

- Failover and load-balancing of EJBs is done with replica-aware stubs.
- Replica-aware stubs are generated at compile time for clusterable EJBs.



ORACLE

18 - 33

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

Clustering EJB Objects: Replica-Aware Stub

Load balancing for clustered EJBs and RMIs is accomplished using the object's replica-aware stub. When a clusterable EJB is compiled, replica-aware stubs are generated for the bean. The replica-aware stub represents a collection of replicas and contains the logic required to locate an EJB or RMI class on any WebLogic Server instance on which the object is deployed.

When a client accesses a clustered object, the replica-aware stub is sent to the client. The stub contains the load-balancing algorithm to balance method calls to the object. On each call, the stub can employ its load algorithm to select a replica. This provides load balancing across the cluster in a way that is transparent to the caller. If a failure occurs during the call, the stub intercepts the exception and retries the call on another replica. This provides failover that is transparent to the caller.

With clustered objects, automatic failover typically occurs only if the object is idempotent. An object is idempotent if any method can be called multiple times with no other effect than calling the method once. This is always true for methods that have no permanent side effects. Methods that have side effects must be written with idempotence in mind.

EJB: Server Failure Situations

A replica-aware stub has to detect an invocation failure from the exceptions it receives:

- Application exceptions
- System exceptions
- Network or communication exceptions

These are not indicative of a critical failure, as your application handles them.

A network exception would occur if a server, container, or skeleton crashed.

Note: If a communication exception occurs, the stub does not know if the method started, was currently executing, or finished but was unable to return a response.



EJB: Server Failure Situations

Because a stub is Java code, it will be able to receive exceptions that are generated by the skeleton, EJB, or the RMI handler (at the network level). Because system and application exceptions are expected, they are not considered failure situations. Application and system exceptions are notifications of abnormalities on the server, but the server is still in a state where it can be used.

If a network or communication exception occurs, this means that the network TCP/IP socket communication with the server has failed. Even though this exception can be the result of a faulty network, it is unlikely. The stub assumes that the server, container, or skeleton has crashed and is temporarily unavailable. Unfortunately, when this scenario occurs, the stub cannot determine the status of the method invocation. The failure may have occurred before, during, or after the method invocation.

Load-Balancing Clustered EJB Objects

- WebLogic Server supports the following load-balancing algorithms for clustered EJB objects:
 - Round-robin
 - Weight-based
 - Random
 - Parameter-based routing (programmatic)
- Server affinity configuration enables calls to objects to remain with the same server and minimizes client-side load balancing.



Load-Balancing Clustered EJB Objects

The following algorithms are supported for clustered EJB objects:

- **Round-Robin (default):** The round-robin algorithm is the default load-balancing strategy for clustered object stubs when no algorithm is specified.
- **Weight-Based:** The weight-based algorithm takes into account a preassigned weight for each server. Each server in the cluster is assigned a weight in the range (1–100). For example, suppose that server A is 4, B is 2, and C is 1, the usage will be ABCABAA... .
- **Random:** This algorithm chooses the next replica at random. This tends to distribute calls evenly among the replicas. It is recommended only in a cluster where each server has the same power and hosts the same services. The advantages are that it is simple and relatively cheap. The primary disadvantage is that there is a small cost to generating a random number on every request, and there is a slight probability that the load will not be evenly balanced over a small number of runs.
- **Parameter-Based Routing:** It is also possible to have a finer grain of control over load-balancing and implemented in the application by the programmer.

Server affinity is accomplished by causing method calls on objects to “stick” to an existing connection, instead of being load-balanced among the available server instances. With server affinity algorithms, the load balancing is disabled only for external client connections.

Stateless Session Bean Failover

A replica-aware stub uses a selection process to implement fault tolerance.

1. Client calls a method on the stub.
2. The stub calls replica-handler to choose server-replica. Load balancing can occur here.
3. The stub calls a method on the replica, (which sends the method to the server).
 - If no exception occurs, the stub returns successfully.
 - If an application or system exception occurs, the stub propagates the exception to the client.
 - If a network or communication exception occurs, the stub calls the replica-handler to choose another replica *if* the method is marked as being idempotent.
 - If a network or communication exception occurs, the stub propagates the exception *if* the method is not marked as being idempotent.



Stateless Session Bean Failover

The algorithm employed in the slide is used by a replica-aware stub on a stateless session bean to choose a replica. Failover should occur when a method that is invoked fails while it is being executed. Because a method can fail in various ways, the different failure situations that can occur need to be analyzed. Depending on the type of exception or failure that is encountered, a stub can react in different ways.

If no exception occurs, the stub will normally return. If a system or application exception is propagated over the wire, the stub does not react to that exception and propagates the exception back to the client. If a network or communication exception occurs, the stub will perform a high availability switch over to another replica if the method is marked as being idempotent. If the method is not marked as being idempotent, the stub just propagates the exception.

Configuring EJB Clustering in Deployment Descriptors

- Clustering of EJBs based on version 2 is configured in the application-specific deployment descriptors.
- When using clustering based on EJB version 3.0, you can use the deployment plans to implement clustering.

A snippet from `weblogic-ejb-jar.xml`:

```
...
<stateless-clustering>
    <stateless-bean-is-clusterable>True
    </stateless-bean-is-clusterable>

    <stateless-bean-load-algorithm>random
    </stateless-bean-load-algorithm>
    ...
</stateless-clustering>
...
```



Configuring EJB Clustering in Deployment Descriptors

When using applications based on EJB 2.x, the cluster parameters are configured in the `weblogic-ejb-jar.xml` or `weblogic-cmp-rdbms-jar.xml` deployment descriptor files. Therefore, WebLogic administrators should discuss with their EJB development team the impact of the clustering features.

EJBs that are based on the 3.0 specification can be configured using annotations and can be configured using deployment plans. However, EJB 3.0 also supports all 2.x WebLogic-specific EJB features, but such features must continue to be configured as per the 2.x WebLogic-specific EJB features in deployment descriptor files.

Configuring EJB Clustering Using the Administration Console

The screenshot shows two panels. On the left is the 'Domain Structure' panel for 'MedRecDomain', with 'Clusters' highlighted (circled in green). A callout '1' points to this selection. On the right is the 'Settings for MedRecCluster' panel, showing the 'General' tab selected. A callout '2' points to the 'Save' button. A callout '3' points to the 'Number Of Servers In Cluster Address:' field, which contains the value '3'. A yellow box with the text 'Default for all EJBs, if not overridden' is positioned over the 'Name:' field ('MedRecCluster'). Another yellow box with the text 'Required only if Address is a single DNS name' is positioned over the 'Cluster Address:' field ('MedRec-System'). A yellow box with the text 'Optional, override the default server addresses and ports used by stubs.' is positioned over the 'Number Of Servers In Cluster Address:' field.

18 - 38

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

ORACLE

Configuring EJB Clustering Using the Administration Console

As an administrator, you configure the default EJB cluster settings for your domain using the following steps:

1. Select Environment > Clusters within the Domain Structure panel of the console. Then select a specific cluster.
2. On the General tab, update any of the fields described as follows:
 - **Default Load Algorithm:** The algorithm used for load balancing between replicated services, such as EJBs, if none is specified for a particular service. The *round-robin* algorithm cycles through a list of Oracle WebLogic Server instances in order. *Weight-based* load balancing improves on the round-robin algorithm by taking into account a preassigned weight for each server. In *random* load balancing, requests are routed to servers at random.
 - **Cluster Address:** The address used by EJB clients to connect to this 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.
 - **Number Of Servers In Cluster Address:** The number of servers listed from this cluster when generating a cluster address automatically

Configuring Clusterable Stateless Session EJBs

- The WLS-specific deployment descriptor has a tag for configuring stateless session EJB clustering parameters.
- A snippet from a typical `weblogic-ejb-jar.xml` file:

```
...
<stateless-session-descriptor>
  <!-- Other Tags As Appropriate Here... -->
  <stateless-clustering>
    <stateless-bean-is-clusterable>True
      </stateless-bean-is-clusterable>
    <stateless-bean-load-algorithm>random
      </stateless-bean-load-algorithm>
    <!-- Other Tags As Appropriate Here... -->
  </stateless-clustering>
...
...
```

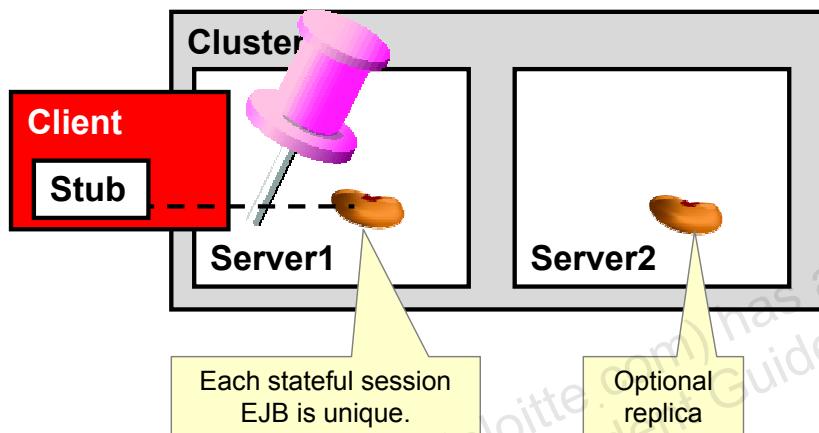


Clusterable Stateless Session Beans

The `<stateless-clustering>` tag specifies options that determine how WebLogic Server replicates stateless session EJB instances in a cluster. When `<stateless-bean-is-clusterable>` is True, the EJB can be deployed from multiple WebLogic Servers in a cluster. Calls to the home stub are load-balanced between the servers on which this bean is deployed, and if a server hosting the bean is unreachable, the call fails over to another server hosting the bean.

Stateful Session Beans

- All calls on a remote stub must be directed to the server that contains the EJB.
- Client is “pinned” to specific stateful session EJB



ORACLE

18 - 40

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

Stateful Session Beans

The relationship between the client and a stateful session bean is as follows:

- Stateful session EJB maintains data unique to each instance.
- To ensure correct data, the stateful session EJB client must use the very same stateful session Bean instance—hence the client is “pinned” to the exact stateful session EJB instance it created.
- Stateful session EJB is managed on a single WebLogic Server in a cluster with an optional (if configured) replica on another server in the same cluster.
- Remote stateful session EJB stub has information embedded about the primary instance of the stateful session EJB location and (if configured) the location of the backup copy—replica.
- Primary stateful session EJB Instance may migrate (if configured) to another server (original replica owner) in the event of the primary owner WebLogic Server outage.
- If stateful session EJB is *not* configured to be clusterable, replication would be disabled and the Remote stub will only have a reference to the Primary owning WebLogic server. In that case, stateful session EJB will not survive the Primary WebLogic server outage.
- Replication behavior is turned on when the stateful session EJB replication is set to **InMemory**, and turned off when set to **None**.

Configuring Clusterable Stateful Session EJBs

- The WLS-specific deployment descriptor has a tag for configuring stateful session EJB clustering parameters.
- The replication type for EJBs is InMemory or None.

```
<stateful-session-descriptor>
  <stateful-session-clustering>
    <home-is-clusterable> true      </home-is-clusterable>
    <home-load-algorithm> random   </home-load-algorithm>
    :
    :
    <replication-type>     InMemory </replication-type>
  </stateful-session-clustering>
</stateful-session-descriptor>
```

The ORACLE logo is displayed on a red horizontal bar.

Configuring Clusterable Stateful Session EJBs

All the tags in `<stateful-session-clustering>` are optional.

Because an instance of a stateful session EJB is connected to a single client, invocations can be sent only to a single server, not a cluster. But the home stub invocations are stateless and thus can be clustered.

`<home-is-clusterable>` indicates if the home stub is clustered.

`<home-load-algorithm>` declares what load balancing algorithm to use.

The `<replication-type>` tag is used to indicate whether or not your stateful session EJB is replicated to a secondary server in the cluster. The value of the `<replication-type>` tag can be `InMemory` or `None`.

Read/Write Versus Read-Only

- There are two types of entity beans to consider:
 - Read/write
 - Read-only
- For read/write entity beans, load balancing and failover occur only at the home level.
- For read-only entity beans, the replica-aware stub:
 - Load balances on every call
 - Does not automatically fail over in the event of a recoverable call failure

ORACLE

18 - 42

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

Read/Write Versus Read-Only

There are two types of entity beans to consider: read/write entities and read-only entities.

- **Read/Write Entities:** When a home finds or creates a read/write entity bean, it obtains an instance on the local server and returns a stub pinned to that server. Load balancing and failover occur only at the home level. Because it is possible for multiple instances of the entity bean to exist in the cluster, each instance must read from the database before each transaction and write on each commit.
- **Read-Only Entities:** When a home finds or creates a read-only entity bean, it returns a replica-aware stub. This stub load-balances on every call but does not automatically fail over in the event of a recoverable call failure. Read-only beans are also cached on every server to avoid database reads.

Entity Bean Cluster-Aware Home Stubs

- Entity beans can have cluster-aware home stubs that have knowledge of the EJB Home objects on all WLS instances in the cluster.
- The `home-is-clusterable` deployment element in the `weblogic-ejb-jar.xml` file determines whether a home stub is cluster aware.
- An example of setting an entity EJB home stub as cluster aware:

```
<entity-clustering>
    <home-is-clusterable>True</home-is-clusterable>
    <home-load-algorithm>random</home-load-algorithm>
    <home-call-router-class-name>beanRouter
    </home-call-router-class-name>
</entity-clustering>
```



Entity Bean Cluster-Aware Home Stubs

In an Oracle WebLogic Server cluster, the server-side representation of the Home object can be replaced by a cluster-aware “stub.” The cluster-aware home stub has knowledge of the EJB Home objects on all the Oracle WebLogic Servers in the cluster. The clustered home stub provides load balancing by distributing EJB lookup requests to available servers. It can also provide failover support for lookup requests because it routes those requests to available servers when other servers have failed.

All EJB types—stateless session, stateful session, and entity EJBs—can have cluster-aware home stubs. Whether or not a cluster-aware home stub is created is determined by the `home-is-clusterable` deployment element in `weblogic-ejb-jar.xml`.

When `home-is-clusterable` is True, the EJB can be deployed from multiple Oracle WebLogic Servers in a cluster. Calls to the home stub are load-balanced between the servers on which this bean is deployed. If a server that hosts the bean is unreachable, the call automatically fails over to another server that hosts the bean.

EJB Best Practices

- Set pool and cache sizes in accordance with anticipated load and execute threads per server.
- Understand that cache sizes equally affect all nodes in the cluster.
- Mark bean methods that can be called multiple times with impunity as idempotent in their deployment descriptors.



EJB Best Practices

Design Idempotent Methods: It is not always possible to determine when a server instance failed with respect to the work it was doing at the time of failure. For instance, if a server instance fails after handling a client request but before returning the response, there is no way to tell that the request was handled. A user that does not get a response retries, resulting in an additional request.

Failover for Remote Method Invocation (RMI) objects requires that methods be *idempotent*. An idempotent method is one that can be repeated with no negative side effects.

To configure idempotence, at bean level, set `stateless-bean-methods-are-idempotent` in `weblogic-ejb-jar.xml` to True. At method level, set `idempotent-methods` in `weblogic-ejb-jar.xml`.

Quiz

Select all valid values for the persistent store type element in weblogic.xml.

- a. file
- b. replicated
- c. unicast
- d. async-replicated-if-clustered
- e. jdbc
- f. async-wan



18 - 45

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

Quiz

Which two Oracle WebLogic Server features can be used to control the destination servers that are used for in-memory replication?

- a. Web service
- b. Replication group
- c. Data source
- d. Node Manager
- e. Machine



18 - 46

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

Quiz

Which is NOT associated with in-memory replication?

- a. Cookie
- b. Secondary
- c. Session
- d. Schema
- e. Primary
- f. Synchronous



18 - 47

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

Quiz

Which types of replication configuration are allowed for EJBs?

- a. JDBC
- b. File
- c. InMemory
- d. None



18 - 48

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

Summary

In this lesson, you should have learned how to:

- Deploy applications to a cluster
- Describe session state in a cluster
- Configure replication groups
- Configure in-memory replication
- Configure JDBC replication
- Configure file replication

Practice 18: Overview Managing Clusters

This practice covers the following topics:

- Defining a cluster as a target for new applications
- Retargeting existing applications to a cluster
- Deploying an application to a cluster
- Setting up in-memory session replication



Practice 18 Overview: Managing Clusters

See Appendix A for the complete steps to do the practice.

19

Security Concepts and Configuration

ORACLE

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

Objectives

After completing this lesson, you should be able to:

- Use the WebLogic Server (WLS) security architecture
- Configure security realms
- Configure users and groups
- Configure roles
- Configure policies
- Configure protection for:
 - Web application resources
 - Enterprise JavaBeans (EJBs)



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

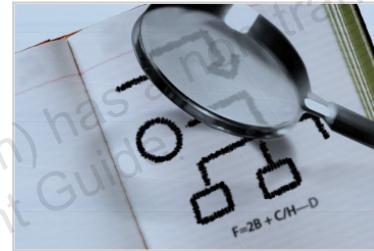
Objectives

Scenario

The Medical Records department has decided to explore the use of the security features provided by Oracle WebLogic Server to protect the application and other resources deployed in the Oracle WebLogic Server domain. You create users, groups, simple authentication, and authorization policies and describe the working of these policies in protecting a typical application.

Road Map

- Security overview
 - Oracle Platform Security Services
 - Oracle WLS Security
 - Oracle WLS Security Models
 - Introduction to WLS Security components
- Users and groups
- Protecting application resources



ORACLE

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

Introduction to Oracle WebLogic Security Service

- Security is a challenge in environments with diverse applications and Web-based services.
- This requires established and well-communicated security policies and procedures.
- You can use Oracle WebLogic Server as a comprehensive and flexible security infrastructure to protect applications.



ORACLE

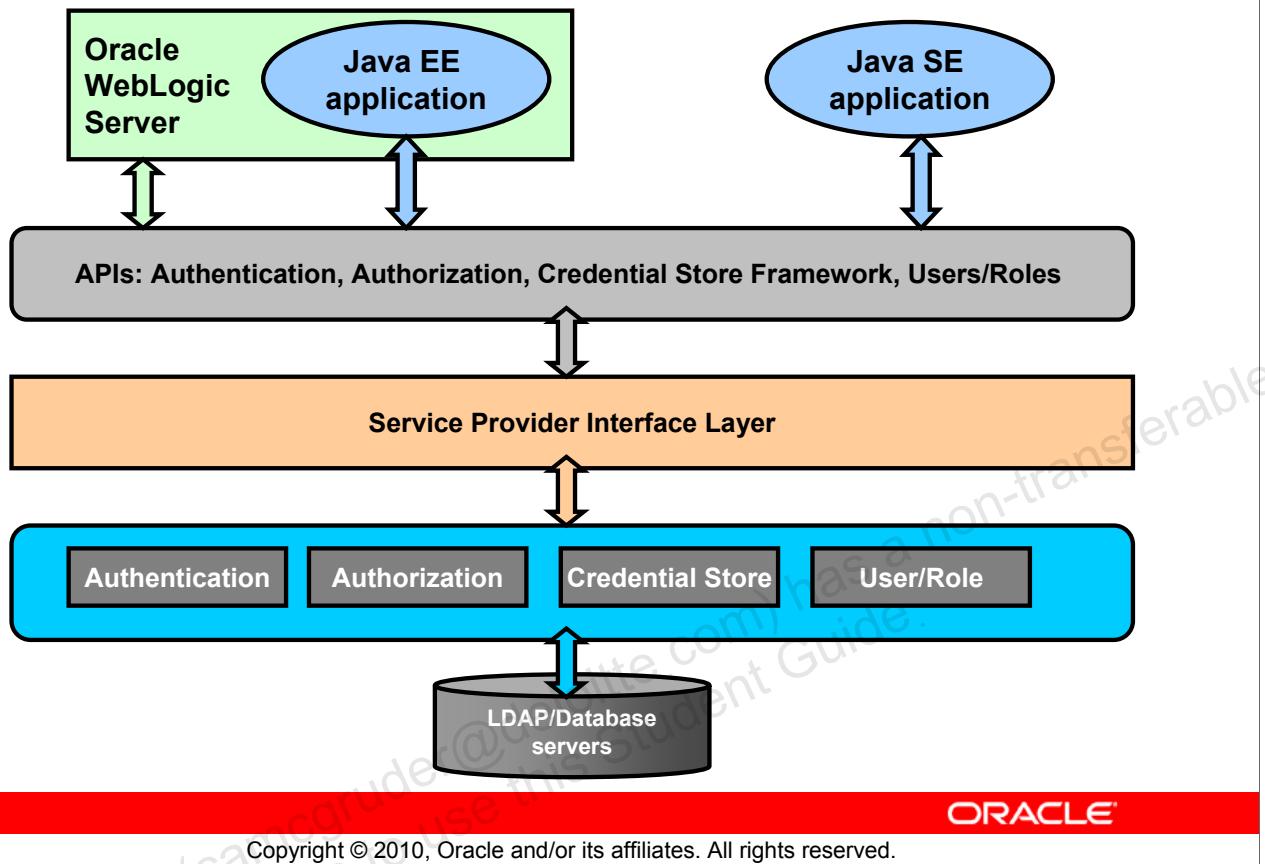
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Introduction to Oracle WebLogic Security Service

Deploying, managing, and maintaining security is a challenge for organizations that provide new and expanded services to customers using the Web. To serve a worldwide network of Web-based users, an organization must address the fundamental issues of maintaining the confidentiality, integrity, and availability of the system and its data. Challenges to security involve every component of the system. Security across the infrastructure requires vigilance as well as established and well-communicated security policies and procedures.

Oracle WebLogic Server includes a security architecture that provides a comprehensive, flexible security infrastructure designed to address the security challenges of making applications available on the Web. WebLogic security can be used stand-alone to secure WebLogic Server applications or as part of an enterprisewide, security management system that represents a best-in-breed, security management solution.

Oracle Platform Security Services



Oracle Platform Security Services

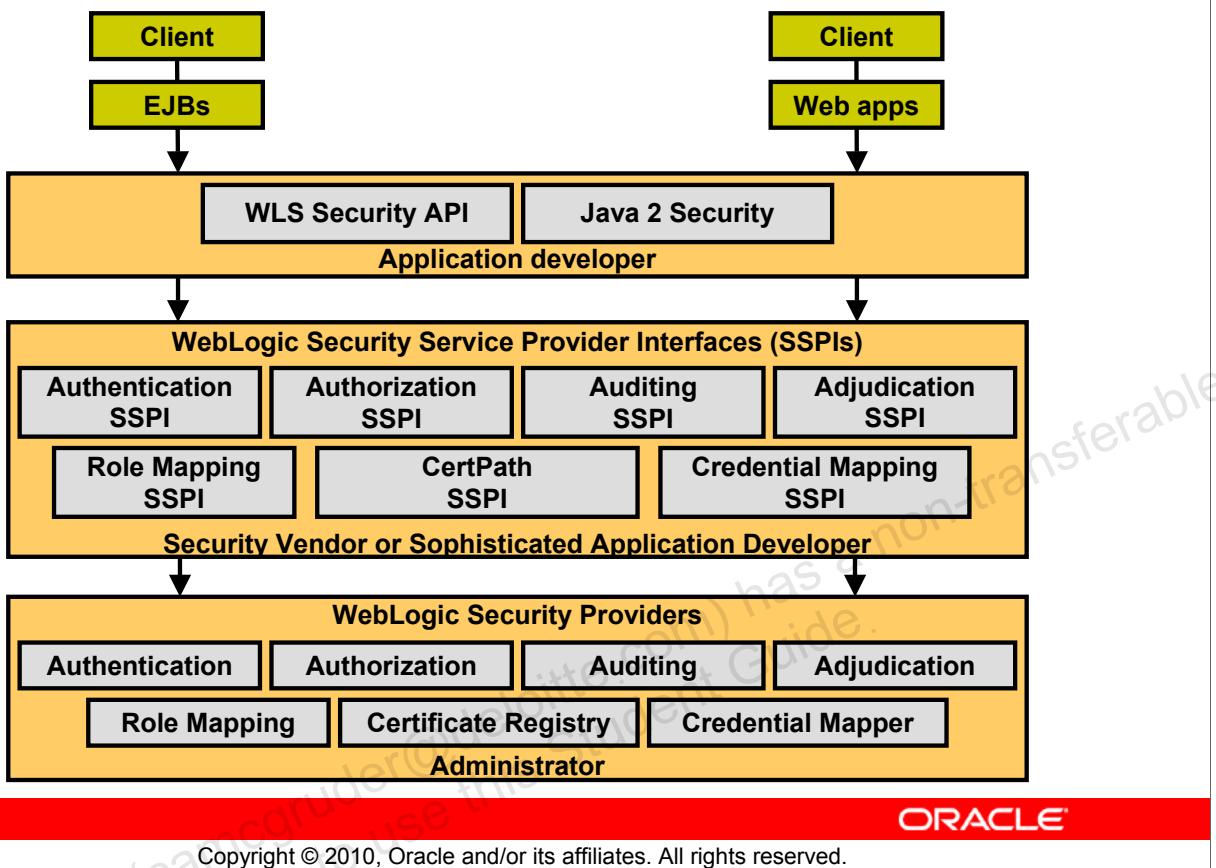
Oracle Platform Security Services (OPSS) is a security framework that runs on Oracle WebLogic Server. It combines the security features of the WebLogic Server and the Oracle Application Server to provide application developers, system integrators, security administrators, and independent software vendors with a comprehensive security platform framework for Java SE and Java EE applications. OPSS offers abstraction layer APIs that insulate developers from security and identity management implementation details.

- Developers can invoke the services provided by OPSS directly from the development environment (Oracle JDeveloper) using wizards.
- Administrators can configure the services of OPSS before and after the application is deployed into the Oracle WebLogic Server using Enterprise Manager pages, the Oracle WebLogic Administration Console, or command-line utilities.

OPSS provides security services to both the Oracle WebLogic Server and to the application deployed on it. Out of the box, Oracle WebLogic Server comes with a part of OPSS referred to as Common Security Services (CSS) that provides security services to the Oracle WLS components. This lesson explains the use of the CSS part of OPSS.

The complete OPSS is available when you install and use other components of Fusion Middleware such as Oracle SOA 11g Suite or Oracle WebCenter 11g Suite, or Oracle JDeveloper Suite. In such installations, you can configure and use OPSS fully. For further information about OPSS, refer to the *Oracle Fusion Middleware Security Guide*.

Oracle WLS Security Architecture



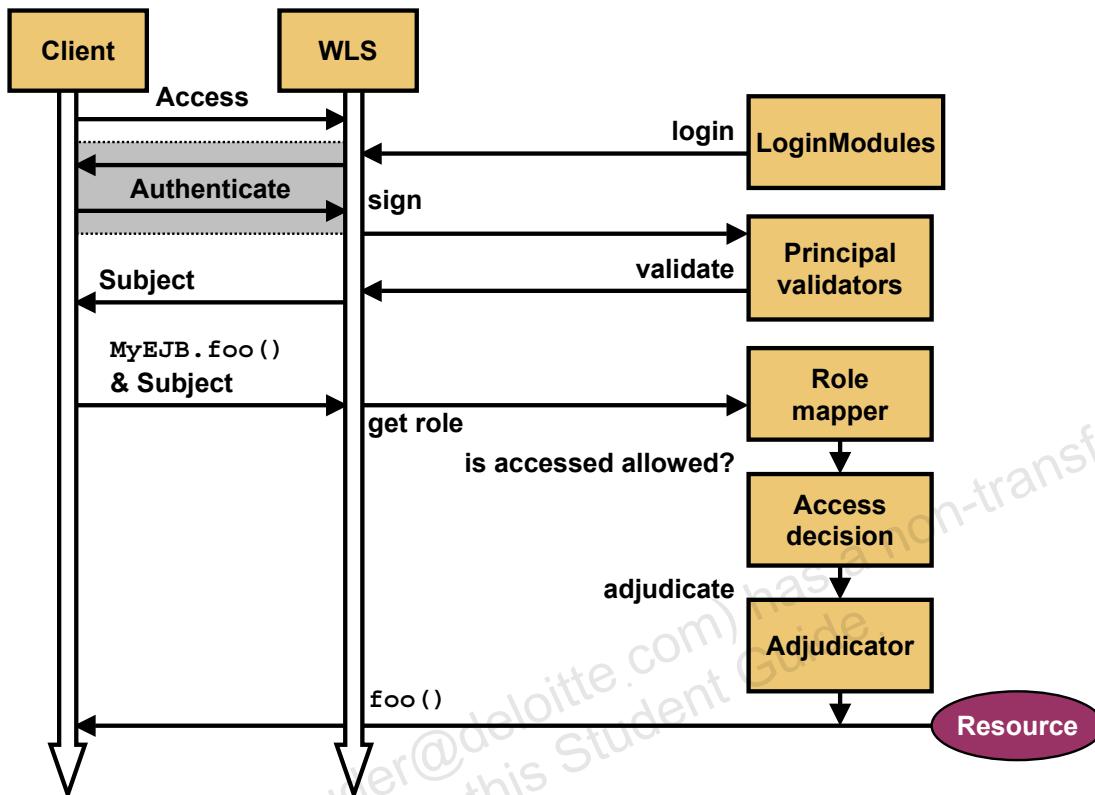
Oracle WLS Security Architecture

The WebLogic Security Service consists of:

1. A set of Security Service Provider Interfaces (SSPIs) for developing new security services that can be plugged into the Oracle WebLogic Server environment. SSPIs are available for Authentication, Authorization, Auditing, Role Mapping, Certificate Lookup and Validation, and Credential Mapping
2. A set of WebLogic security providers. These security providers are the Oracle implementation of the SSPIs and are available by default in the Oracle WebLogic Server product. The WebLogic security providers include Authentication, Authorization, and Auditing.
3. A set of APIs that allow application developers to specify authorization information that is used when Oracle WebLogic Server acts as a client and to obtain information about the Subject and Principals used by Oracle WebLogic Server
4. J2SE 5.0 security packages, including Java Secure Socket Extensions (JSSE), Java Authentication and Authorization Service (JAAS), Java Security Manager, Java Cryptography Architecture and Java Cryptography Extensions (JCE), and Java Authorization Contract for Containers (JACC)

For more information, refer to the *Oracle Fusion Middleware Understanding Security for Oracle WebLogic Server* documentation.

Security Services



ORACLE

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

Security Services

In a simple authentication, a user (or a client application), also referred to as the *subject*, attempts to log in to a system with a username/password combination. Oracle WebLogic Server establishes trust by validating that user's username and password. A principal represents the subject and the subject's features or properties. A subject can contain multiple principals. When the user (subject) enters the name and password, these properties and any other related information are encapsulated into the principal.

The validation of a principal is performed by the principal validator. After successfully proving the subject's identity, an authentication context is established, which allows an identified user or system to be authenticated to other entities.

During the authorization process, Oracle WebLogic Server determines whether a given subject can perform a given operation on a given resource, and returns the result of that decision to the client application. This process requires the use of access decisions, an adjudication provider, and possibly multiple role mapping providers.

Roles are obtained from the Role Mapping providers and input to the Access Decisions. The Access Decisions are then consulted for an authorization result. If multiple Access Decisions are configured and return conflicting authorization results (such as PERMIT and DENY), an Adjudication provider is used to resolve the contradiction by returning a final decision.

Overview of Security Concepts

- Authentication providers handle identity information and make it possible to associate with users, groups, or roles.
- Identity assertion providers map a valid token to an Oracle WebLogic Server user.
- An authorization provider is a process that is used to control the interactions between users and resources based on user identity.
- The adjudication provider weighs the results that multiple access decisions return to determine the final decision.
- The credential mapping process is initiated when application components access the authentication mechanism of a legacy system to obtain a set of credentials.
- Auditing provides a trail of activity. The auditing provider is used to log activity before and after security operations.



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

Overview of Security Concepts

Authentication is the mechanism to answer the question “Who are you?” using credentials such as username/password combinations to determine whether the caller is acting on behalf of specific users or system processes. In WLS, authentication providers prove the identity of users or system processes and transport and make identity information available to the components of a system (via subjects) when needed.

A LoginModule authenticates the password and stores principals into the subject. There is a one- to-one relationship between an authentication provider and a LoginModule. Each authentication provider’s LoginModule store principals into the same subject.

Authorization answers the question “What can you access?” based on user identity or other information. Oracle WebLogic Server provides an authorization provider to limit the interactions between users and WebLogic resources to ensure integrity, confidentiality, and availability.

Authorization providers use access decision components to answer the question “Is access allowed?” Can a subject perform an operation on a WebLogic resource with specific parameters in an application? The result is PERMIT, DENY, or ABSTAIN.

Oracle WebLogic Server provides an auditing provider to collect, store, and distribute information about requests and the outcome of those requests for nonrepudiation. You can configure multiple auditing providers in a security realm, but none are required.

Confidentiality

- Oracle WebLogic Server supports the secure sockets layer (SSL) protocol to secure the communication between the clients and the server.
- The SSL client authentication allows a server to confirm a user's identity by verifying that a client's certificate and public ID are valid and are issued by a Certificate Authority (CA).
- The SSL server authentication allows a user to confirm a server's identity by verifying that the server's certificate and public ID are valid and are issued by a CA.

ORACLE

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

Confidentiality

Oracle WebLogic Server supports the SSL protocol to enable secure communication between the applications that are connected through the Web. By default, WebLogic Server is configured for one-way SSL authentication where the managed server is enabled with a digital certificate. Using the Administration Console, you can configure Oracle WebLogic Server for two-way SSL authentication where the client and server are both enabled with digital certificates to securely establish their identity.

To use SSL, you would require a private key, a digital certificate containing the matching public key, and a certificate signed by at least one trusted CA to verify the data embedded in the digital certificate. For intermediate authorities, you may need to install the root-trusted CA's certificate.

SSL server authentication allows a user to confirm a server's identity, through an SSL-enabled client software using standard techniques of public key cryptography, to verify that a server's certificate and public ID are valid and have been issued by a CA that is listed in the client's list of trusted CAs. For example, when sending a credit card number, you may want to check the receiving server's identity.

Confidentiality (continued)

SSL client authentication allows a server to confirm a user's identity to verify that a client's certificate and public ID are valid and have been issued by a CA that is listed in the server's list of trusted CAs. For example, if a bank sends the account information to a customer, this check may be essential.

The SSL protocol includes two subprotocols: the SSL record protocol, which defines the format that is used to transmit data, and the SSL handshake protocol to exchange a series of messages between an SSL-enabled server and an SSL-enabled client when the SSL connection is established.

Credential Mapping

- The credential mapping process is used when application components access the authentication mechanism of an external system to obtain a set of credentials.
- The requesting application passes the subject as part of the call and information about the type of credentials required.
- Credentials are returned to the security framework, which is then passed to the requesting application component.
- The application component uses the credentials to access the external system.



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

Credential Mapping

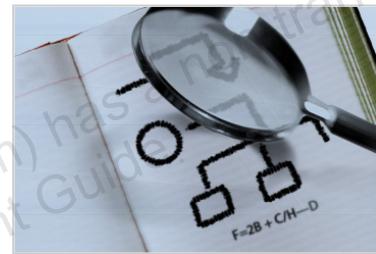
A credential map is a mapping of credentials used by Oracle WebLogic Server to credentials that are used in a legacy (or a remote) system to connect to a given resource in that system. Credential maps allow Oracle WebLogic Server to log in to a remote system on behalf of a subject that has already been authenticated.

A credential mapping provider of WLS can handle several different types of credentials, such as username/password, Kerberos tickets, and public key certificates. Credential mappings can be set in deployment descriptors or through the Administration Console.

You can configure multiple credential mapping providers in a security realm. The security framework makes a call to each credential mapping provider to determine whether it contains the type of credentials requested by the container. The framework accumulates and returns all the credentials as a list.

Road Map

- Security overview
- Users and groups
 - Security realms
 - Configuring users, groups, and roles
 - Embedded LDAP
- Protecting application resources

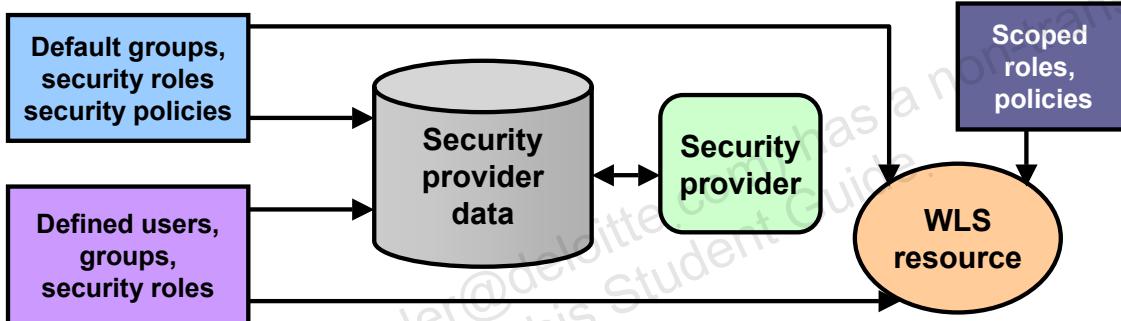


ORACLE

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

Security Realms

- A security realm is a collection of system resources and security service providers.
- A valid user must be authenticated by the authentication provider in the security realm.
- Only one security realm can be active at a given time.
- A single security policy can be used in any realm.
- Administration tasks include creating security realms.



ORACLE

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

Security Realms

A security realm is a mechanism for protecting Oracle WebLogic Server resources, such as authenticators, adjudicators, authorizers, auditors, role mappers, and credential mappers. Oracle WebLogic Server resources in a domain are protected under only one security realm and by a single security policy in that security realm. A user must be defined in a security realm in order to access any resources belonging to that realm. When a user attempts to access a particular Oracle WebLogic Server resource, Oracle WebLogic Server tries to authenticate the user and then authorize the user action by checking the access privileges that are assigned to the user in the relevant realm.

Security Model Options for Applications

Security Model	Location of Users, Roles, and Policies	Security Checks Performed
Deployment Descriptor Only (Java EE standard)	Deployment descriptors: <ul style="list-style-type: none">• web.xml and weblogic.xml• ejb-jar.xml and weblogic-ejb-jar.xml	Only when clients request URLs or EJB methods that are protected by a policy in the deployment descriptor
Custom Roles	Role mappings from a role mapping provider that you configure for the security realm Policies are defined in the web.xml and ejb-jar.xml deployment descriptors.	Only when clients request URLs or EJB methods that are protected by a policy in the deployment descriptor.
Custom Roles and Policies	Role mappings and authorization from providers that you configure for the security realm	For all URLs and EJB methods in the application
Advanced	This model is fully flexible. You can import security data from deployment descriptors into the security provider databases to provide a baseline.	Configurable

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

Security Model Options for Applications

You choose a security model when you deploy each Web application or EJB, and your choice is immutable for the lifetime of the deployment. If you want to use a different model, you must delete and redeploy the Web application or EJB.

The Java EE platform already provides a standard model for securing Web applications and EJBs. In this standard model, you define role mappings and policies in the deployment descriptors of Web application or EJB.

Because this Java EE standard can be too inflexible for some environments, WebLogic Server offers a choice of other, more flexible models in addition to supporting the Java EE standard.

Security Model Options for Applications (continued)

Deployment Descriptor Only model

- This is the standard Java EE model and is therefore a widely known technique for adding declarative security to Web applications and EJBs.
- It uses only roles and policies defined by a developer in the Java EE deployment descriptor (DD) and the WebLogic Server DD.
- It requires the security administrator to verify that the security principals (groups or users) in the deployment descriptors exist and are mapped properly in the security realm.
- With this model, EJBs and URL patterns are not protected by roles and policies of a broader scope (such as a policy scoped to an entire Web application). If an EJB or URL pattern is not protected by a role or policy in the DD, it is unprotected: anyone can access it.
- This model is appropriate if developers and security administrators can closely coordinate their work, both upon initial deployment of the Web application or EJB and upon subsequent redeployments.

Custom Roles model

- The model enables team members to focus on their areas of expertise. Web application and EJB developers need only to declare which URL patterns or EJB methods should be secured. Then the security administrator creates role mappings that fit within the existing hierarchy of roles and principals for a given realm.
- If a developer changes policies in a deployment descriptor, WebLogic Server recognizes the change as soon as you redeploy the Web application or EJB. If an administrator changes role mappings, the changes take effect immediately without requiring a redeployment.

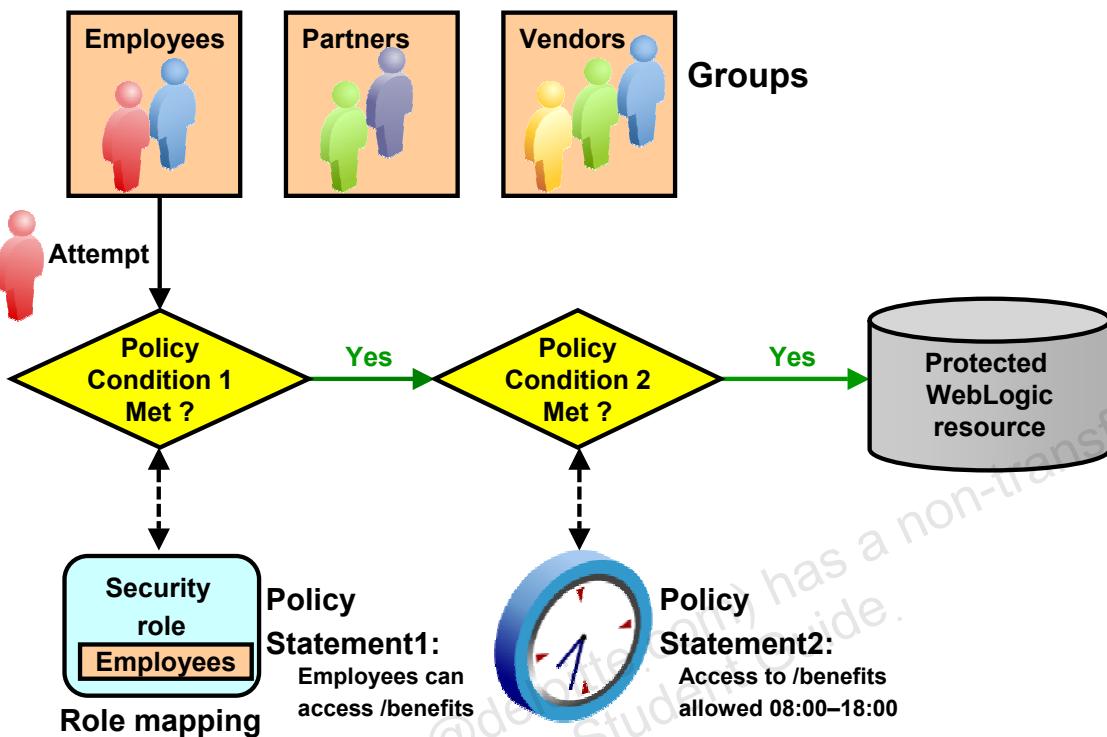
Custom Roles and Policies model

- This security model offers unified and dynamic security management. Instead of requiring developers to modify multiple deployment descriptors when organizational security requirements change, administrators can modify all security configurations from a centralized graphical user interface.
- Users, groups, security roles, and security policies can all be defined using the Administration Console. As a result, the process of making changes based on updated security requirements becomes more efficient.
- This model is appropriate if you require only that entire Web applications or EJBs be secured, but is less appropriate if you require fine-grained control of a large number of specific URL patterns or EJB methods.

Advanced model

- WebLogic Server provides this model primarily for backwards compatibility with releases prior to 9.0.

How WLS Resources Are Protected



ORACLE

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

How WLS Resources Are Protected

The following steps provide an overview of the process of granting access to a WLS resource:

- As an administrator, before creating security policies and roles, you can create users and groups and statically assign users to groups that represent organizational boundaries.
- Then create a security role based on your established business procedures. The security role consists of one or more conditions that specify the circumstances under which a particular user, group, or other role should be granted the security role.
- At run time, the WebLogic Security Service compares the groups against the role conditions to determine whether users in the group should be dynamically granted a security role. This process of comparing groups to roles is called role mapping.
- Then, you create a security policy based on your established business procedures. The security policy consists of one or more policy conditions that specify the circumstances under which a particular security role should be granted access to a WebLogic resource.
- At run time, the WebLogic Security Service uses the security policy to determine whether access to the protected WebLogic resource should be granted. Only users who are members of the group that is granted the security role can access the WebLogic resource.

Users and Groups

- Users are entities that use WLS, such as:
 - Application end users
 - Client applications
 - Other Oracle WebLogic Servers
- Groups are:
 - Logical sets of users
 - More efficient for managing a large number of users



ORACLE

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

Users and Groups

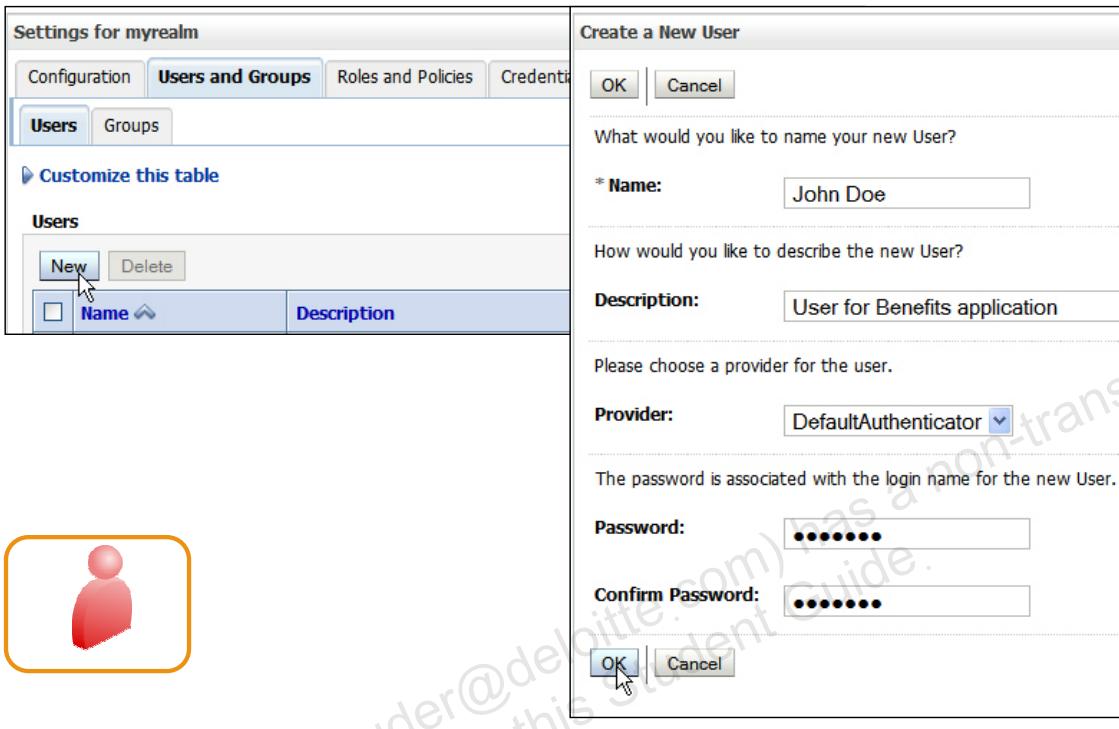
Users are entities that can be authenticated in a security realm. A user can be a person, such as an application end user, or a software entity, such as a client application, or other instances of WebLogic Server. As a result of authentication, a user is assigned an identity or principal. Each user is given a unique identity within the security realm.

Users may be placed into groups that are associated with security roles or be directly associated with security roles.

When users want to access WebLogic Server, they present proof material (such as a password or a digital certificate) to the authentication provider configured in the security realm. If WebLogic Server can verify the identity of the user based on that username and credential, WebLogic Server associates the principal assigned to the user with a thread that executes code on behalf of the user. Before the thread begins executing code, however, WebLogic Server checks the security policy of the WebLogic resource and the principal (that the user has been assigned) to make sure that the user has the required permissions to continue.

A person can be defined as both an individual user and a group member. Individual-access permissions override any group member-access permissions. Oracle WebLogic Server evaluates each user by first looking for a group and testing whether the user is a member of the group and then looking for the user in the list of defined users.

Configuring New Users



ORACLE

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

Configuring New Users

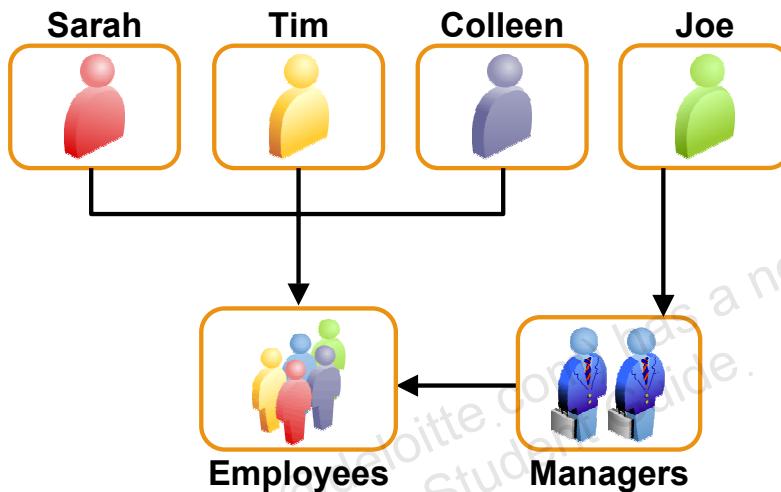
To configure a new user, perform the following steps:

1. Access Security Realms and select your security realm in the Realms Table on the Summary of Security Realms page.
2. Click the Users and Groups > Users tab for your realm. Click New in the Users table.
3. Enter the necessary details in the “Create a New User” dialog box and click OK. The name may contain spaces, but other systems may not allow spaces. Best practice is to reserve the use of spaces for the description and use underscores if needed for the name.

Groups

WLS provides the flexibility to organize groups in various ways:

- Groups can contain users.
- Groups can contain other groups.



ORACLE

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

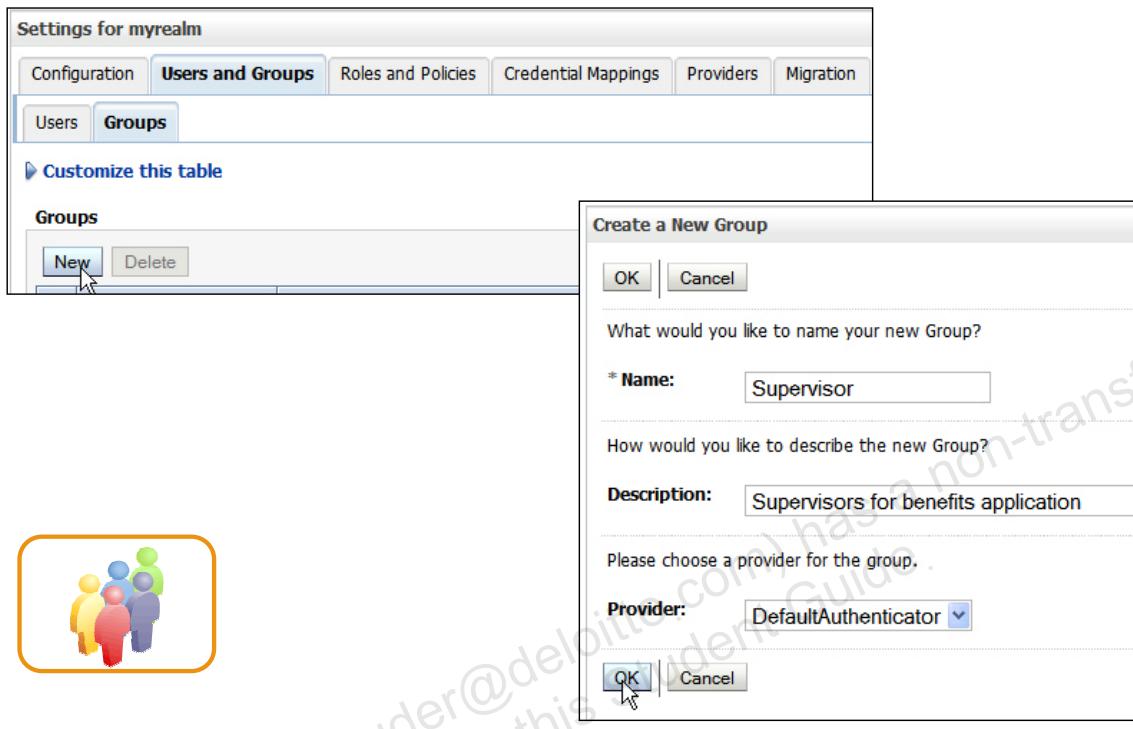
Groups

Groups can be organized in arbitrary ways, thereby providing greater flexibility. In this example, all the users (Sarah, Tim, Colleen, and Joe) are members of the Employees group. Joe is also a member of the Managers group. All Managers are also Employees.

Managing groups is more efficient than managing large numbers of users individually. For example, an administrator can specify permissions for 50 users at one time if those 50 users belong to the same group. Usually, group members have something in common. For example, a company may separate its sales staff into two groups: Sales Representatives and Sales Managers. This is because staff members have different levels of access to the Oracle WebLogic Server resources depending on their job descriptions.

Oracle WebLogic Server can be configured to assign users to groups. Each group shares a common set of permissions that govern its member users' access to resources. You can mix group names and usernames whenever a list of users is permitted.

Configuring New Groups



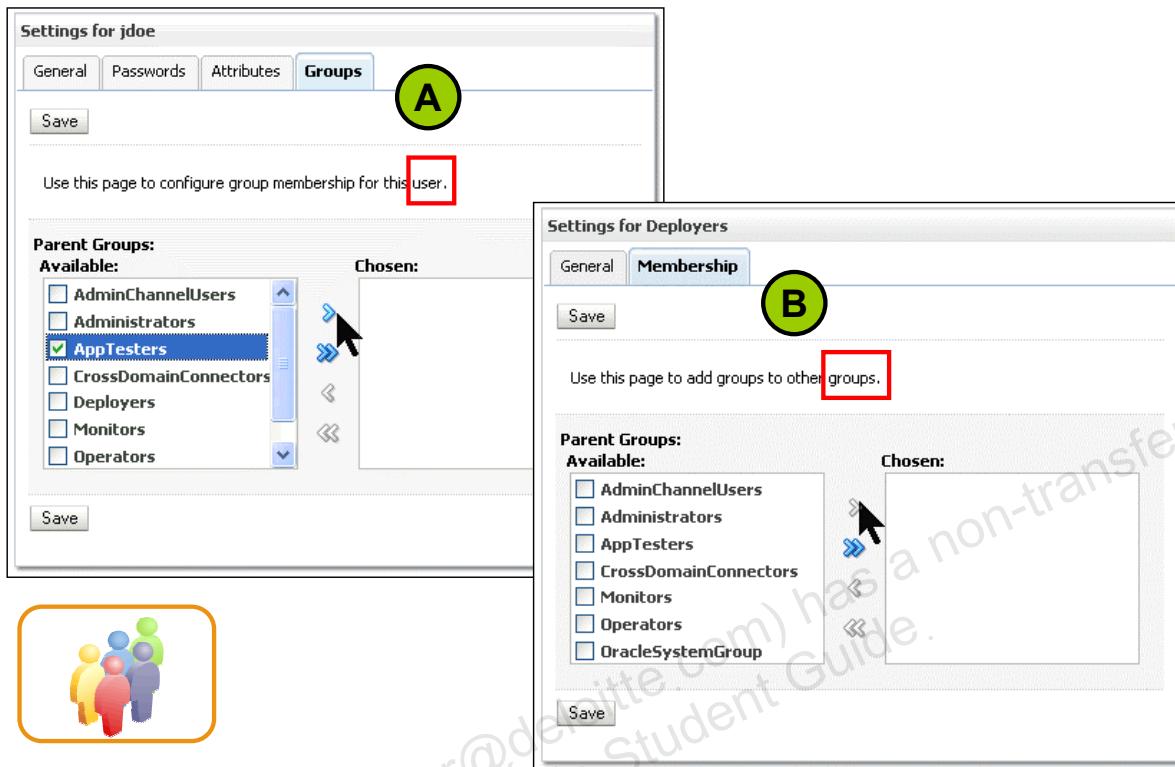
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Configuring New Groups

To configure a new group, perform the following steps:

1. Access Security Realms and select your security realm in the Realms Table on the Summary of Security Realms page.
2. Click the Users and Groups > Groups tab for your realm. Click New in the Groups table.
3. Enter the necessary details in the “Create a New Group” dialog box and click OK.

Configuring Group Memberships



ORACLE

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

Configuring Group Memberships

Each group has two types of membership.

- You can configure a user to be a member of a group as follows:
 - Navigate to the Users subtab under the Users and Groups tab of the security realm.
 - Select the user for whom you want to configure the group membership.
 - Click the Groups tab on the “Settings for the <user>” page.
 - Select the group from the Available list and click > to move it to the Chosen list. Then click Save.
- You can configure a group to be a member of another group as follows:
 - Navigate to the Groups subtab under the Users and Groups tab of the security realm.
 - Select the Group you want to configure as a child of another group.
 - Click the Membership tab on the “Settings for the <group>” page.
 - Select the parent group from the Available list and click > to move it to the Chosen list. Then click Save.

Embedded LDAP Server

- In WLS, users, groups, and authorization information can be stored in an embedded LDAP server.
- Several properties can be set to manage the LDAP server, including:
 - Credentials
 - Backup settings
 - Cache settings
 - Replication settings



ORACLE

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

Embedded LDAP Server

The embedded LDAP server is used as a storage mechanism with the Oracle WebLogic Server authentication, authorization, role mapping, and credential mapping providers.

Information from these providers is stored and updated in the administration server and replicated to all the managed servers in the domain. The read operations performed by the Oracle WebLogic Server security providers (when running on a managed server) access the local replicated embedded LDAP server. The write operations access the master embedded LDAP server on the administration server and any updates are replicated to all the managed servers in the domain. If the administration server is not running, operations by the Oracle WebLogic Server security providers that write to the embedded LDAP server (for example, adding new users, groups, or roles, or adding resources) are not possible.

Configuring an Embedded LDAP

This page allows you to configure the embedded LDAP server for this WebLogic Server domain.

<input type="text"/> Credential:	*****	<input checked="" type="checkbox"/> Cache Enabled
<input type="text"/> Confirm Credential:	*****	<input type="text"/> Cache Size: 32
<input type="text"/> Backup Hour:	23	<input type="text"/> Cache TTL: 60
<input type="text"/> Backup Minute:	5	<input type="checkbox"/> Refresh Replica At Startup
<input type="text"/> Backup Copies:	7	<input type="checkbox"/> Master First
		<input type="text"/> Timeout: 0
		<input type="checkbox"/> Anonymous Bind Allowed

Save

ORACLE

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

Configuring an Embedded LDAP

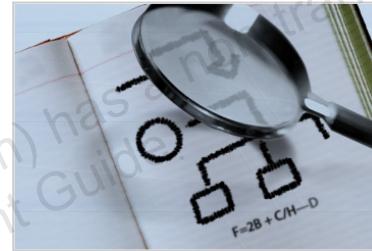
- **Credential:** The credential (usually password) that is used to connect to the embedded LDAP server. This password is encrypted. The default is null.
- **Backup Hour:** The hour at which to back up the embedded LDAP server. Minimum is 0, Maximum is 23, and Default is 23.
- **Backup Minute:** The minute at which to back up the embedded LDAP server. This attribute is used with the BackupHour attribute to determine the time at which the embedded LDAP server is backed up. Minimum is 0, Maximum is 59, and Default is 05.
- **Backup Copies:** The number of backup copies of the embedded LDAP server. Minimum is 0, Maximum is 65534, and Default is 7.
- **Cache Enabled:** Whether or not a cache is used for the embedded LDAP server. The default is True.
- **Cache Size:** The size of the cache (in KB) that is used with the embedded LDAP server. Minimum is 0 and Default is 32.
- **Cache TTL:** The time-to-live (TTL) of the cache in seconds. Minimum is 0 and Maximum is 60.

Configuring an Embedded LDAP (continued)

- **Refresh Replica At Startup:** Whether or not a managed server should refresh all replicated data at boot time. This is useful if you made a large number of changes when the managed server was not active and you want to download the entire replica instead of having the administration server push each change to the managed server. The default is false.
- **Master First:** The connections to the master LDAP server should always be made instead of connections to the local replicated LDAP server. The default is false.

Road Map

- Security overview
- Users and groups
- Roles and policies
 - Security roles
 - Security policies
 - Defining policies and roles
 - Protecting Web resources
 - Protecting other resources



ORACLE

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

Security Roles

- A role refers to a set of privileges granted to a user or group.
- A role differs from a group; a group has static membership, whereas a role is conditional.
- A user and group can be granted multiple roles.
- The two types of roles are:
 - Global-scoped roles
 - Resource-scoped roles
- Global roles include Admin, Operator, Deployer, Monitor, AppTester, Anonymous, and others.
- Roles defined in deployment descriptors can be inherited.
- You can manage role definitions and assignments without editing deployment descriptors or redeploying the application.

ORACLE

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

Security Roles

A security role is a privilege granted to users or groups based on specific conditions. Similar to groups, security roles allow you to restrict access to WebLogic resources for several users simultaneously. However, unlike groups, security roles:

- Are evaluated and granted to users or groups dynamically, based on conditions such as username, group membership, or the time of day
- Can be scoped to specific WebLogic resources within an application in a WebLogic Server domain (unlike groups, which are always scoped to an entire WebLogic Server domain)

Granting a security role to a user or a group confers the defined access privileges to that user or group as long as the user or group is “in” the security role. Multiple users or groups can be granted a single security role. A role definition is specific to a security realm.

A role can be defined as global or scoped.

WLS defines a set of default global roles for protecting all the WebLogic resources in a domain. A scoped role protects a specific resource, such as a method of an EJB or a branch of the JNDI tree. Most roles are scoped. The default global roles are: Admin, AdminChannelUser, Anonymous, AppTester, CrossDomainConnector, Deployer, Monitor, Operator, and OracleSystemRole.

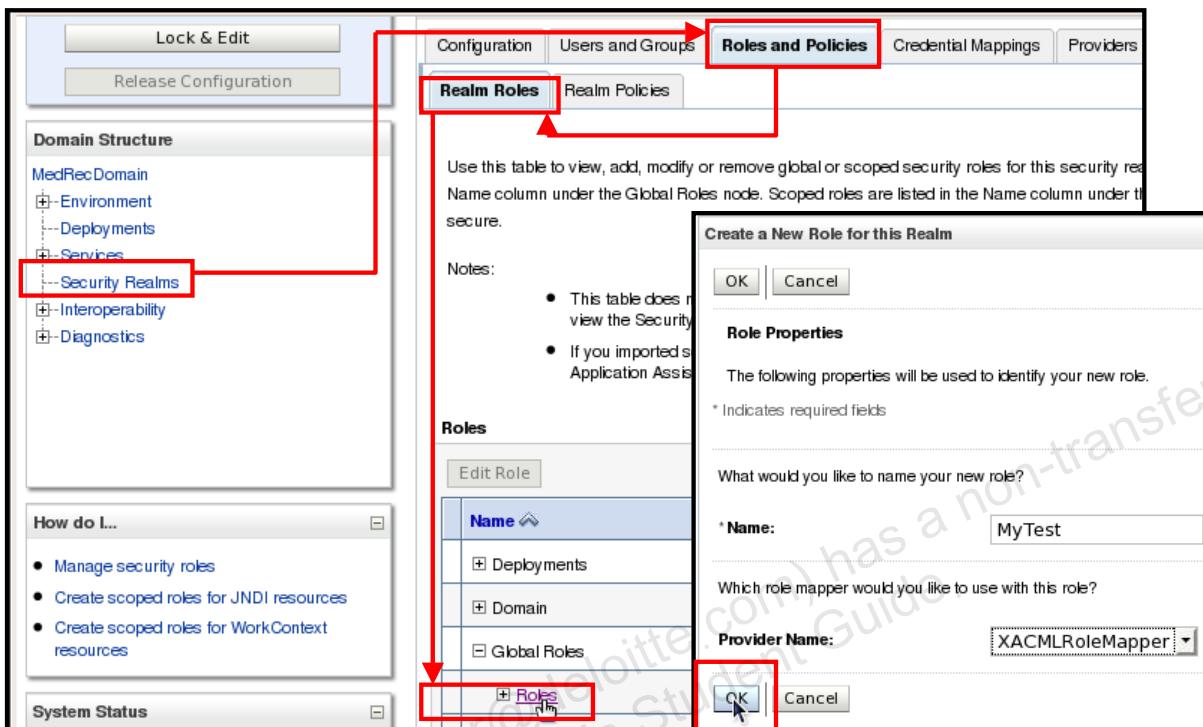
Note that by default no security role is enforced and therefore all the resources can be accessed by any user.

Security Roles (continued)

Default Global Roles Provided by Oracle WebLogic Server

- **Admin** can display and modify all resource attributes and perform start and stop operations. By default, users in the Administrators group are granted the Admin role. You can change this association or add other group associations.
- **Operator** can display all resource attributes. Users can start, suspend, resume, and stop resources. By default, users in the Operators group are granted the Operator role. You can change this association or other group associations.
- **Deployer** can display all resource attributes. Users can deploy applications, EJBs, and other deployable modules. By default, users in the Deployers group are granted the Deployer role. You can change this association or other group associations.
- **Monitor** can display all resource attributes. Users can modify the resource attributes and operations that are not restricted to the other roles. By default, users in the Monitors group are granted this role. You can change this association or other group associations.
- **AppTester** can test the versions of applications that are deployed to Administration mode.
- **Anonymous:** All users are granted this global role.

Configuring the Global Security Role



ORACLE

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

Configuring the Global Security Role

To create a global security role:

- In the left pane of the Administration Console, select Security Realms. On the “Summary of Security Realms” page, select the name of the realm in which you want to create the role (for example, myrealm).
- On the Settings page, click the Roles and Policies tab. Then click the Roles subtab. The Roles page organizes all the domain’s resources and corresponding roles in a hierarchical tree control.
- In the Roles table, in the Name column, expand the Global Roles node. In the Name column, select the name of the Roles node.
- In the Global Roles table, click New. On the “Create a New Role for this Realm” page, enter the name of the global role in the Name field.
- If you have more than one role mapper configured for the realm, from the Provider Name list, select the role mapper you want to use for this role. Click OK to save your changes.
- In the Global Roles table, select the role. In the Role Conditions section, click Add Conditions.
- On the “Choose a Predicate” page, in the Predicate List, select a condition.
- The next steps depend on the condition that you chose. After you complete the conditions, click Finish.

Security Policies

- Security policies implement parameterized authorization.
- Security policies comprise rules and conditions.
- Users and groups that adhere to the security policy are granted access to resources protected by the policy.
- Security policies follow a hierarchy. The policy of a narrower scope overrides that of a broader scope.
- When you install Oracle WebLogic Server, some default root-level policies are provided.



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

Security Policies

Oracle WebLogic Server provides security policies and roles as two mechanisms that are used together to control access to or protect resources. The security realm that Oracle WebLogic Server provides stores policies in the embedded LDAP server.

You can create a root-level policy that applies to all instances of a specific resource type. For example, you can define a root-level policy that applies to all JMS resources in your domain.

You can also create a policy that applies to a specific resource instance. If the instance contains other resources, the policy will apply to the included resource as well. For example, you can create a policy for an entire Enterprise Archive (EAR), an EJB JAR containing multiple EJBs, a particular EJB within that JAR, or a single method within that EJB.

The policy of a narrower scope overrides the policy of a broader scope. For example, if you create a security policy for an EAR and a policy for an EJB that is in the EAR, the EJB will be protected by its own policy and will ignore the policy for the EAR.

Policy Conditions

- Policy conditions are the essential components of a policy.
- The WebLogic Server authorization provides three kinds of built-in policy conditions in the Administration Console:
 - Basic policy conditions
 - Date and Time policy conditions
 - Context Element policy conditions

ORACLE

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

Policy Conditions

To determine who can access a resource, a policy contains one or more conditions. The most basic policy simply contains the name of a security role or a principal. For example, a basic policy might simply name the global role Admin. At run time, the WebLogic Service interprets this policy as “allow access if the user is in the Admin role.”

You can create more complex conditions and combine them using the logical operators AND and OR (which is an inclusive OR). You can also negate any condition, which would prohibit access under the specified condition.

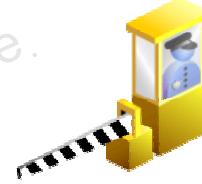
WebLogic Server by default provides three kinds of conditions:

- **Basic:** This can be used to allow or deny access to everyone or specific users, groups or roles.
- **Date and Time:** When you use any of the date and time conditions, the security policy grants access to all users for the date or time you specify, unless you further restrict the users by adding one of the other conditions.
- **Context Element:** You can use the context element conditions to create security policies based on the value of HTTP Servlet Request attributes, HTTP Session attributes, and EJB method parameters. WebLogic Server retrieves this information from the ContextHandler object and allows you to define policy conditions based on the values. When using any of these conditions, it is your responsibility to ensure that the attribute or parameter/value pairs apply to the context in which you are using them.

Protecting Web Applications

To protect a Web application with declarative security, perform the following steps:

1. Define the roles that should access the protected resources.
2. Determine the Web application resources that must be protected.
3. Map the protected resources to roles that should access them.
4. Map roles to users or groups in the WLS security realm.
5. Set up an authentication mechanism.



ORACLE

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

Protecting Web Applications

If you are using the DD Only or Custom Roles security model for the deployment of a Web application, you cannot use the Administration Console to modify its security policies. You have to define your security details using the deployment descriptors.

Specifying Protected Web Resources

Protection for Web resources are based on URL patterns.

The screenshot shows the 'Settings for benefits(Red)' page with the 'Security' tab selected. Under 'Application Scope > URL Patterns', the 'Policies' subtab is active. A 'Stand-Alone Web Application URL Patterns' table is displayed, showing a single row with a 'New' button highlighted by a green circle labeled '1'. A modal dialog titled 'Create a New Stand-Alone Web Application URL Pattern Scoped Policy' is open, containing fields for 'URL Pattern' (set to '/managers/*') and 'Provider Name' (set to 'XACMLAuthorizer'). The 'OK' button in the dialog is also highlighted by a green circle labeled '2'.

Example URL patterns:

URL Pattern	Role Name
/*	Some role name (for example, director)
/*.jsp	employee
/EastCoast/*	east-coaster

ORACLE

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

Specifying Protected Web Resources

URL patterns provide a flexible way to define security for a single resource or a group of resources.

- In the Administration Console, navigate to your domain > Deployments and click the Web application in the Deployments table.
- On the Settings for *<the application>* page, navigate to Security > Application Scope > URL Patterns (subtab). Click New in the Stand-Alone Web Application URL Pattern Scoped Roles table.
- Specify the URL pattern (for example, /managers/*) and then specify the name: director. This is the name that the application has been configured to use in securing its resources. Leave the provider name as XACMLRoleMapper and click OK.
- In the Stand-Alone Web Application URL Pattern Scoped Roles table, you should now see the URL pattern created.
- Click URL Pattern and click Add Conditions. Choose the appropriate predicate from the Predicate List and click Next. On the next page, enter the appropriate conditions and values and then click Add.
- Click Finish. On the next page, click Save.

Defining Policies and Roles for Other Resources

You can define roles and policies for other resources, such as JDBC and JMS.

The screenshot shows the 'Settings for myDataSource' page. The 'Security' tab is selected (circled 2). In the 'Policy Conditions' section, the 'Add Conditions' button is highlighted (circled 3). A modal dialog titled 'Choose a Predicate' is open, showing a list of predicates. The 'Next' button is highlighted (circled 4). Another modal dialog titled 'Edit Arguments' is open, showing time settings. The 'Finish' button is highlighted (circled 5).

Summary of JDBC Data Sources

A JDBC data source is an object bound to the JNDI through a pool of JDBC connections. Applications can then borrow a database connection from a data source.

This page summarizes the JDBC data source object.

Customize this table

Data Sources (Filtered - More Columns Exist)

Click the **Lock & Edit** button in the Change Center to activate all the buttons on this page.

	Name	JNDI Name	Targets
<input type="checkbox"/>	MedRecGlobalDataSourceXA	java:comp/env/jdbc/MedRecGlobalDataSourceXA	
<input type="checkbox"/>	myDataSource	java:comp/env/jdbc/myDataSource	

Choose a Predicate

Choose the predicate you wish to use as your new condition.

Predicate List: Access occurs between specified times.

Edit Arguments

On this page you will fill in the arguments that pertain to this predicate.

The earliest permissible time in the format of "h:mm AM|PM".

Starting time: 08:00:00 AM

The latest permissible time in the format of "h:mm AM|PM".

Ending time: 07:59:59 PM

The time ahead of GMT (enter as "GMT+h:mm") or behind Eastern Standard Time in the USA would be "GMT-5:00".

GMT offset: GMT-8:00

Buttons: Back, Next, Finish, Cancel

ORACLE

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

Defining Policies and Roles for Other Resources

Defining roles and policies for other resources is similar to defining roles and policies for the Web resources. For all of them, you need to define policy conditions and policy statements. For some resources, you can also define methods or actions that are allowed for that resource. For instance, for servers, you may define restrictions on actions such as boot, shutdown, lock, and unlock.

The following steps illustrate how you can define a policy for the myDataSource JDBC data source:

1. In the Administration Console, navigate to Services > JDBC > Data Sources. In the Data Sources table, click the data source for which you want to define policy.
2. On the Settings for <resource> (myDataSource) page, select Security > Policies.
3. Click Add Conditions in the Policy Conditions section.
4. Select the appropriate choice from the Predicate List and click Next.
5. Specify the appropriate conditions and click Finish.

Configuring Authentication

Configure how a Web application determines the security credentials of users:

- **BASIC**: The Web browser displays a dialog box.
- **FORM**: Use a custom HTML form.
- **CLIENT-CERT**: Request a client certificate.

Configure the authentication using the `<login-config>` element:

```
:<login-config>
    <auth-method>BASIC, FORM, or CLIENT-CERT</auth-method>
    <form-login-config>
        <form-login-page>login.jsp</form-login-page>
        <form-error-page>badLogin.jsp</form-error-page>
    </form-login-config>
</login-config>
```



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

Configuring Authentication

Configure how users will be authenticated in your Web application using the `<login-config>` element. J2EE provides three types of authentication:

- **BASIC**: A Web browser is used to display a dialog box with fields for a username and password.
- **FORM**: A specified HTML page, JSP, or servlet is used to display an HTML form with the username and password text fields. The generated form must conform to a set of specifications. Use the `<form-login-config>` element to specify the resource that contains the form. The `<form-error-page>` element defines the JSP, servlet, or HTML file to display if the user's credentials are invalid.
- **CLIENT-CERT**: WebLogic Server may receive digital certificates as part of Web Services requests, two-way SSL, or other secure interactions. To validate these certificates, WebLogic Server includes a Certificate Lookup and Validation (CLV) framework, whose function is to look up and validate X.509 certificate chains. The key elements of the CLV framework are CertPathBuilder and CertPathValidators. The CLV framework requires one and only one active CertPathBuilder which, given a reference to a certificate chain, finds the chain and validates it, and zero or more CertPathValidators which, given a certificate chain, validates it.

Authentication Examples

BASIC authentication



FORM-based authentication



ORACLE

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

Authentication Examples

Oracle WebLogic Server supports three types of authentication for Web browsers:

- **BASIC**
- **FORM**
- **CLIENT - CERT**

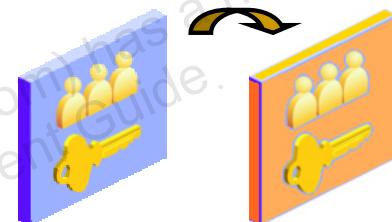
With **BASIC** authentication, the Web browser displays a dialog box in response to a WebLogic resource request. The login screen prompts the user for a username and password. The slide shows a typical login screen.

When using **FORM** authentication with Web applications, you provide a custom login screen that the Web browser displays in response to a Web application resource request and an error screen that displays if the login fails. The login screen can be generated using an HTML page, JSP, or servlet. The benefit of **FORM**-based login is that you have complete control over these screens. You can design them to meet the requirements of your application or enterprise policy or guideline.

The login screen prompts the user for a username and password.

Migrating Security Data

- You can export users and groups, security policies, security roles, or credential maps between security realms or domains.
- It is useful, for example, in transitioning from development to QA to production.
- You can use migration constraints (key/value pairs) to specify the export/import options.
- Currently, the system supports migrating only security data between the WLS security providers.



ORACLE

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

Migrating Security Data

Oracle WebLogic Server security realms persist different kinds of security data—for example, users and groups (for the WebLogic authentication provider), security policies (for the XACML authorization provider), security roles (for the XACML role mapping provider), and credential maps (for the WebLogic credential mapping provider).

When you configure a new security realm or a new security provider, you may prefer to use the security data from your existing realm or provider, rather than re-create all the users, groups, policies, roles, and credential maps. Several WebLogic security providers support security data migration. This means that you can export security data from one security realm and import it into a new security realm. You can migrate security data for each security provider individually or migrate security data for all the WebLogic security providers simultaneously (that is, security data for an entire security realm).

Note that you can migrate security data from one provider to another only if the providers use the same data format. You migrate security data through the WebLogic Administration Console or by using the WebLogic Scripting Tool (WLST).

Migrating Security Data (continued)

Migrating security data may be helpful when you:

- Transition from development to production mode
- Copy production mode security configurations to security realms in the new Oracle WebLogic Server domains
- Move data from one security realm to a new security realm in the same Oracle WebLogic Server domain, where one or more of the default WebLogic security providers are replaced with new security providers

Exporting the WLS Default Authenticator Provider

The screenshot shows the Oracle WebLogic Server Administration Console interface. On the left, under 'Settings for myrealm', the 'Providers' tab is active. In the 'Authentication Providers' section, there are three entries: 'DefaultAuthenticator' and 'DefaultIdentityAsser' (partially visible). On the right, under 'Settings for DefaultAuthenticator', the 'Migration' tab is active. The 'Export' button is highlighted with a red box. Below the tabs, a note says: 'This page allows you to export security data from this security provider's database to a file.' A save button is also present. At the bottom, there is a red bar with the 'ORACLE' logo and a copyright notice: 'Copyright © 2010, Oracle and/or its affiliates. All rights reserved.'

Exporting the WLS Default Authenticator Provider

To export security data from a security provider to a file, perform the following steps:

1. In the left pane, select Security Realms and then select the name of the realm that you are configuring (for example, `myrealm`).
 2. Select the type of provider from which you want to export the security data (for example, authentication).
 3. Select the security provider from which you want to export the security data.
 4. Select Migration > Export.
 5. Specify the directory and file name in which to export the security data in the "Export File on Server" field. The directory must exist.
- Note:** The directory and file into which you export the security data should be carefully protected with operating system security because they contain secure information about your deployment.
6. Optionally, define a specific set of security data to be exported in the Export Constraints (`key=value`) box.
 7. Click Save.

Note: After the data is exported from the security provider, it can be imported at any time.

Importing into a Different Domain

The screenshot shows the 'Settings for myrealm' interface. In the top navigation bar, the 'Providers' tab is selected. Below it, the 'Authentication' tab is also selected. The 'Authentication Providers' section lists three entries: 'DefaultAuthenticator' (selected) and 'DefaultIdentityAsser'. On the right, the 'Settings for DefaultAuthenticator' page is displayed with the 'Migration' tab selected. The 'Import' button is highlighted with a red box and a mouse cursor. Below it is the 'Save' button. A descriptive text explains the import process: 'This page allows you to import security data from a file to this security provider's database.' A note indicates that fields marked with an asterisk (*) are required. The 'Import Format' dropdown is set to 'DefaultAtn'. The 'Import File on Server' field contains the path '/u01/app/work/domains/MedRecDomain/DefaultAuth...'. The 'Supported Import Constraints' field shows 'None'. An empty 'Import Constraints (key=value)' input field is present. To the left of the main interface, there is a graphic of a blue cylinder (database) with a blue arrow pointing towards it, and a purple card with icons of people and a key.

ORACLE

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

Importing into a Different Domain

You can export the security data from a security provider into a file and then import the data into a different security provider. As an alternative, you can export the security data from all the security providers in a realm and then import that data into another security realm. To import security data into a security provider, perform the following steps:

1. In the left pane of the Administration Console, select Security Realms.
2. Select the name of the security realm into which the security data is to be imported (for example, `myrealm`).
3. Select Providers and then the type of provider into which the security data is to be imported (for example, Providers > Authentication).
4. Select the security provider into which the security data is to be imported and select Migration > Import.
5. Specify the directory and file name of the file that contains the exported security data in the "Import File on Server" field.
6. You can restrict the imported security parameters by specifying the Import Constraints.
7. Click Save.

Summary

In this lesson, you should have learned how to:

- Use the WLS security architecture
- Configure users, groups, and roles
- Configure roles
- Configure policies
- Configure protection for:
 - Web application resources
 - EJBs
- Configure security realms



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

Practice 19: Overview

Configuring Security for WLS Resources

This practice covers the following topics:

- Creating new users using the Administration Console
- Creating groups of employees and managers
- Assigning groups to users
- Configuring groups-to-role mapping
- Defining resources that are protected by the security you have configured
- Verifying that the security protection that you enabled is working



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

Practice 19 Overview: Configuring Security for WLS Resources

See Appendix A for the complete steps to do the practice.

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

Carl McGruder (camcgruder@deloitte.com) has a non-transferable
license to use this Student Guide.

20

Protecting Against Attacks

ORACLE®

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

Objectives

After completing this lesson, you should be able to:

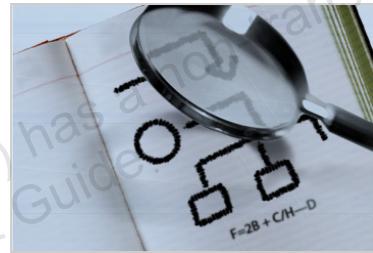
- Describe the process of configuring secure sockets layer (SSL)
- Use the keytool utility to configure keys and obtain digital certificates
- Configure SSL for the WLS server
- Configure countermeasures for some Web-based attacks such as:
 - Man in the middle
 - Denial of service
 - Large buffer
 - Connection starvation



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

Road Map

- Protecting the transport layer
 - SSL
 - keytool
 - Certificates
 - Configuring SSL
- Protecting against attacks



ORACLE

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

What Is SSL?

SSL is a protocol that enables:

- Connection security through encryption
- A server to authenticate to a client
- A client to authenticate to a server (optional)
- Data integrity such that the data that flows between a client and server is protected from tampering by a third party



ORACLE

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

What Is SSL?

The SSL protocol offers security to applications that are connected through a network. Specifically, the SSL protocol provides the following:

- A mechanism that the applications can use to authenticate each other's identity
- Encryption of the data that is exchanged by the applications
- Data integrity, whereby the data that flows between a client and a server is protected from tampering by a third party

When the SSL protocol is used, the target always authenticates itself to the initiator. Optionally, if the target requests it, the initiator can authenticate itself to the target. Encryption makes the data that is transmitted over the network intelligible only to the intended recipient. An SSL connection begins with a handshake during which time the applications exchange digital certificates, agree on the encryption algorithms to be used, and generate the encryption keys to be used for the remainder of the session.

Trust and Identity

- SSL and keystore are configured independently.
- For the purpose of backward compatibility, this release of Oracle WebLogic Server supports private keys and a trusted WebLogic Keystore provider.
- Identity:
 - Private key and digital certificate (Can now be looked up directly from the keystore, not necessarily as a stand-alone file outside the keystore)
- Trust:
 - Certificates of trusted certificate authorities



ORACLE

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

Trust and Identity

For demonstration purposes, you can use the following out of the box:

<WL_HOME>\server\lib\DemoIdentity.jks as identity, and
<WL_HOME>\server\lib\DemoTrust.jks or
<JAVA_HOME>\jre\lib\security\cacerts for trust.

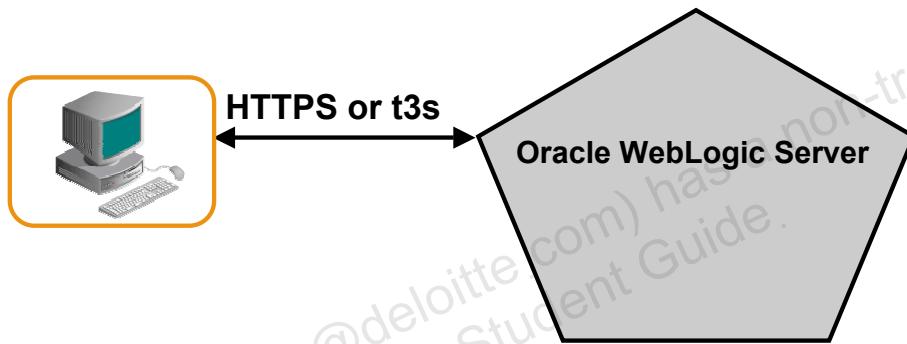
By default the Node Manager and server SSL use DemoTrust.jks for trust.

To create identity and trust for a server, perform the following steps:

1. Obtain digital certificates, private keys, and trusted CA certificates from the CertGen utility, Sun Microsystems' keytool utility, or a reputable vendor such as Entrust or VeriSign. You can also use the digital certificates, private keys, and trusted CA certificates provided by the Oracle WebLogic Server kit. The digital certificates, private keys, and trusted CA certificates in the demonstration should be used only in a development environment.
2. Store the private keys, digital certificates, and trusted CA certificates. Private keys and trusted CA certificates are stored in a keystore.
Note: The preferred keystore format is Java KeyStore (JKS). Oracle WebLogic Server supports private keys and trusted CA certificates that are stored in files or in the WebLogic Keystore provider only for the purpose of backward compatibility.
3. Configure the identity and trust keystores for Oracle WebLogic Server in the Oracle WebLogic Server Administration Console.

Using an SSL Connection

- WLS uses SSL to secure HTTP and t3 communication.
- To use SSL, clients access WLS via the HTTPS or t3s protocols.
 - `https://localhost:7002/orderStock`
 - `t3s://localhost:7002/useCreditCard`



ORACLE

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

Using an SSL Connection

The use of SSL is signified in the protocol scheme of the URL to specify the location of Oracle WebLogic Server. SSL communications between Web browsers and Oracle WebLogic Server are encapsulated in HTTPS packets for transport. For example:

`https://myserver.com:7002/mypage.html`

Oracle WebLogic Server supports HTTPS with Web browsers that support SSL version 3. Java clients connect to Oracle WebLogic Server with the SSL protocol tunnel over Oracle's multiplexed t3 protocol. For example:

`t3s://myserver.com:7002`

Java clients running in Oracle WebLogic Server can also establish either t3s connections to other Oracle WebLogic Servers, or HTTPS connections to other servers that support the SSL protocol, such as Web servers or secure proxy servers. Browsers connect securely to Oracle WebLogic Server by specifying the appropriate protocol (that is, HTTPS) in the requested URL, whereas Java clients have a variety of options available to them when setting up secure connections. Java clients can use the SSL libraries in Oracle WebLogic Server to provide the SSL socket or, alternatively, they can use an SSL provider such as Sun Microsystems' Java Secure Socket Extension (JSSE) as the SSL socket.

Using an SSL Connection (continued)

When using Oracle WebLogic Server's SSL libraries, the client can create an HTTPS connection directly by using WLS-specific classes, such as `weblogic.net.http.HttpsURLConnection`. This connection can then be used to send and receive secure data as with any other `java.net.HttpURLConnection`.

Clients can also use JNDI to set up an SSL connection (for example, to an EJB). This can be done by specifying a t3s connection within `PROVIDER_URL` and "strong" as the `SECURITY_AUTHENTICATION` type when populating the Hashtable object that is used to create the JNDI InitialContext.

Finally, clients can use another SSL provider's implementation to set up a secure connection with WLS. Sun Microsystems' JSSE implementation is a popular choice. JSSE has been integrated into the Java 2 SDK, Standard Edition (J2SDK), v 1.4. It is a collection of Java packages that allow for secure Internet communications. It is a Java implementation of the SSL and Transport Layer Security (TLS) protocols that allow for encryption, authentication (both server and client), and message integrity. After the client imports the proper packages and initializes the JSSE service, it uses the standard `java.net.HttpURLConnection` to create a secure connection.

Enabling Secure Communication

- With SSL, data is encrypted using a negotiated symmetric session key.
- A public key algorithm is used to negotiate the symmetric session key.
- In SSL, digital certificates are used to provide a trusted public key.



ORACLE

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

Enabling Secure Communication

Under normal, non-Internet circumstances, data is sent between two parties. Each party has the same key and can decipher the data. Such situations in which both parties use the same key to encrypt and decrypt the data are termed *symmetric key encryption*. The problem with symmetric key encryption is that anyone can potentially see anything that is transmitted over the Internet by intercepting its key as it is being transferred.

When you use public key/private key encryption, the public key is freely available and can be transferred across the Internet. Anyone can use the public key. Data is encrypted with the public key, but can be decrypted only with the private key, which is held privately in secure storage. Though the two keys are mathematically linked, it is statistically impossible to generate the private key programmatically, thus ensuring data security.

Typically, anyone who wants to send an encrypted message obtains a digital certificate from a trusted source known as a *Certificate Authority* or CA. The CA issues a digital certificate containing the applicant's public key and identification information. The digital certificate is then encrypted by the CA whose own public key is publicly available. The receiver of the message uses the CA's public key to decode the digital certificate attached to the message, verifies it, and then obtains the sender's public key and identification information that is held within the certificate. With this information, the recipient can send an encrypted reply, which only the originator can decrypt.

Enabling Secure Communication (continued)

Using the sender's public and private keys, a symmetric key is established between the communicating parties and eventually secure communication is achieved using a symmetric algorithm. The symmetric key is valid only for the duration of the connection, thus making symmetric key guessing very difficult.

Oracle WebLogic Server SSL Requirements

To enable Oracle WebLogic Server SSL, you must perform the following steps:

1. Obtain an appropriate digital certificate.
2. Install the certificate.
3. Configure SSL properties.
4. Configure two-way authentication (if desired).
 - SSL impacts performance.



ORACLE

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

keytool Utility

- keytool is a standard J2SE SDK utility for managing:
 - The generation of private keys and the corresponding digital certificates
 - Keystores (databases) of private keys and the associated certificates
- The keytool utility can display certificate and keystore contents.
- Specify an algorithm different from DSA when generating digital keys using keytool.



ORACLE

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

keytool Utility

The Sun Microsystems' keytool utility can also be used to generate a private key, a self-signed digital certificate for Oracle WebLogic Server, and a Certificate Signing Request (CSR). Submit the CSR to a certificate authority to obtain a digital certificate for Oracle WebLogic Server.

You can use the keytool utility to:

- Update the self-signed digital certificate with a new digital certificate
- Obtain trust and identity when using Oracle WebLogic Server in a production environment

For more information about Sun's keytool utility, see the keytool Key and Certificate Management Tool description at <http://java.sun.com/j2se/1.5.0/docs/tooldocs/windows/keytool.html>.

Note: When you use the keytool utility, specify an algorithm different from the default Digital Signature Algorithm (DSA) such as RSA because Oracle WebLogic Server does not support DSA.

Obtaining a Digital Certificate: keytool Examples

JDKs include a command-line tool to create, view, and modify keystore files.

- Generate a new self-signed certificate and private key and add it to a store:

```
keytool -genkeypair -alias mykey -keypass mykeypass
    -keyalg RSA -keysize 512 -dname "CN=payroll.mycompany.com..."
    -keystore mykeys.jks -storepass mypass
```

- Import a signed certificate reply from a CA:

```
keytool -importcert -file payroll.pem -alias mykey -keypass
    mykeypass -keystore mykeys.jks -storepass mypass
```

- Inspect the contents of a store:

```
keytool -list -v -keystore mykeys.jks -storepass mypass
```



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

Obtaining a Digital Certificate: keytool Examples

`keytool` is a key and certificate management utility. It allows users to administer their own public/private key pairs and associated certificates for use in self-authentication (where the user authenticates himself/herself to other users/services) or data integrity and authentication services, using digital signatures. It also allows users to cache the public keys (in the form of certificates) of their communicating peers. `keytool` also enables users to administer secret keys used in symmetric encryption/decryption (for example, DES). `keytool` stores the keys and certificates in a keystore.

Available commands include:

- `-genkeypair`: Generates a key pair (a public key and associated private key). Wraps the public key into an X.509 v3 self-signed certificate, which is stored as a single-element certificate chain. This certificate chain and the private key are stored in a new keystore entry identified by `alias`. `keyalg` specifies the algorithm to be used to generate the key pair, and `keysize` specifies the size of each key to be generated. `sigalg` specifies the algorithm that should be used to sign the self-signed certificate; this algorithm must be compatible with `keyalg`. `dname` specifies the X.500 Distinguished Name to be associated with `alias`, and is used as the issuer and subject fields in the self-signed certificate. If no distinguished name is provided at the command line, the user will be prompted for one.

Obtaining a Digital Certificate: `keytool` Examples (continued)

- `-importcert`: Reads the certificate or certificate chain (where the latter is supplied in a PKCS#7 formatted reply or a sequence of X.509 certificates) from the file `cert_file`, and stores it in the keystore entry identified by `alias`. If no file is given, the certificate or certificate chain is read from standard in (“standard input” or “standard input stream”).
`keytool` can import X.509 v1, v2, and v3 certificates, and PKCS#7 formatted certificate chains consisting of certificates of that type. The data to be imported must be provided either in binary encoding format, or in printable encoding format (also known as Base64 encoding) as defined by the Internet RFC 1421 standard. In the latter case, the encoding must be bounded at the beginning by a string that starts with `-----BEGIN`, and bounded at the end by a string that starts with `-----END`. If the alias does not point to a key entry, `keytool` assumes you are adding a trusted certificate entry. In this case, the alias should not already exist in the keystore.
- If the alias does already exist, `keytool` outputs an error, because there is already a trusted certificate for that alias, and does not import the certificate. If the alias points to a key entry, `keytool` assumes you are importing a certificate reply. When importing a certificate reply, the certificate reply is validated using trusted certificates from the keystore, and optionally using the certificates configured in the `cacerts` keystore file (if the `-trustcacerts` option was specified). If the reply is a single X.509 certificate, `keytool` attempts to establish a trust chain, starting at the certificate reply and ending at a self-signed certificate (belonging to a root CA). The certificate reply and the hierarchy of certificates used to authenticate the certificate reply form the new certificate chain of alias. If a trust chain cannot be established, the certificate reply is not imported. In this case, `keytool` does not print out the certificate and prompt the user to verify it, because it is very hard (if not impossible) for a user to determine the authenticity of the certificate reply.
- `-list`: Prints to standard out the contents of the keystore entry identified by `alias`. If no alias is specified, the contents of the entire keystore are printed. This command by default prints the MD5 fingerprint of a certificate. If the `-v` option is specified, the certificate is printed in human-readable format, with additional information such as the owner, issuer, serial number, and any extensions. If the `-rfc` option is specified, certificate contents are printed using the printable encoding format, as defined by the Internet RFC 1421 standard.
- `-delete`: Deletes from the keystore the entry identified by `alias`. The user is prompted for the alias, if no alias is provided at the command line.

Configuring Keystores

Settings for MedRecSrv1

Configuration Protocols Logging Debug Monitoring Control

General Cluster Services **Keystores** SSL Federation Service

Overload Health Monitoring Server Start

Keystores: Demo Identity and Demo Trust **Change**

Identity

Demo Identity Keystore: /u01/app/oracle/Middleware/11.1.1/wlserver_10.3/server/lib/DemoIdentity.jks

Demo Identity Keystore Type: jks

Demo Identity Keystore Passphrase: *****

Trust

Demo Trust Keystore: /u01/app/oracle/Middleware/11.1.1/wlserver_10.3/server/lib/DemoTrust.jks

Demo Trust Keystore Type: jks

Demo Trust Keystore Passphrase: *****

Java Standard Trust Keystore: /u01/app/oracle/Middleware/11.1.1/jrockit_160_17_R28.0.0-679/jre/lib/security/cacerts

Java Standard Trust Keystore Type: jks

Java Standard Trust Keystore Passphrase: *****

Confirm Java Standard Trust Keystore Passphrase: *****

Save

ORACLE

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

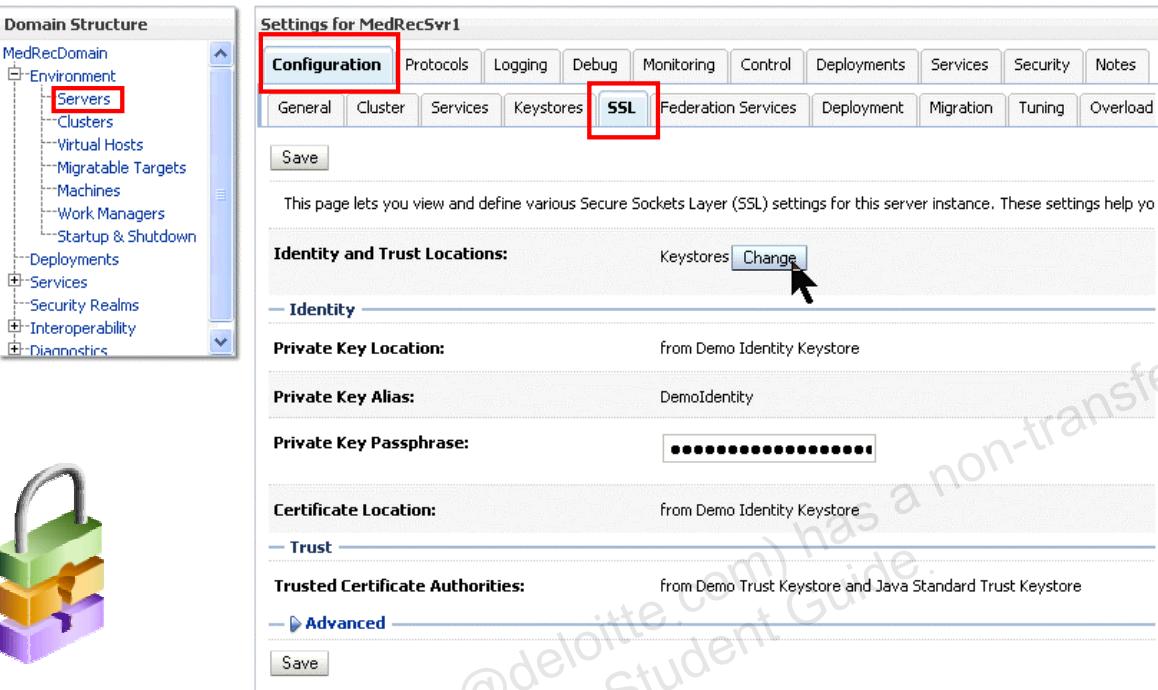
Configuring Keystores

Keystores ensure the secure storage and management of private keys and trusted CAs. WebLogic Server is configured with a default identity keystore (`DemoIdentity.jks`) and a default trust keystore (`DemoTrust.jks`). In addition, WebLogic Server trusts the CA certificates in the JDK `cacerts` file. This default keystore configuration is appropriate for testing and development purposes. However, these keystores should not be used in a production environment.

After you configure identity and trust keystores for a WebLogic Server instance, you can configure its SSL attributes. These attributes include information about the identity and trust location for particular server instances.

For purposes of backward compatibility, with WebLogic Server, you can store private keys and trusted certificates authorities in files or in the WebLogic Keystore provider. If you use either of these mechanisms for identity and trust, select the Files or Keystore Providers (Deprecated) option on the Configuration: SSL page.

Configuring SSL for an Oracle WebLogic Server



The screenshot shows the Oracle WebLogic Administration Console interface. On the left, there is a tree view of the 'Domain Structure' under 'MedRecDomain'. The 'Servers' node is selected and highlighted with a red box. On the right, the 'Settings for MedRecSvr1' window is open. The 'Configuration' tab is selected and highlighted with a red box. Within the 'Configuration' tab, the 'SSL' sub-tab is also highlighted with a red box. A mouse cursor is hovering over the 'Change' button next to the 'Keystores' section. The interface includes tabs for Protocols, Logging, Debug, Monitoring, Control, Deployments, Services, Security, and Notes. Below the tabs, there is a 'Save' button and a descriptive text about SSL settings. The 'Identity and Trust Locations' section contains fields for Private Key Location (from Demo Identity Keystore), Private Key Alias (DemoIdentity), and Private Key Passphrase (redacted). The 'Certificate Location' field also points to the Demo Identity Keystore. The 'Trusted Certificate Authorities' section indicates it uses the Demo Trust Keystore and Java Standard Trust Keystore. At the bottom, there is an 'Advanced' section and another 'Save' button. A watermark across the page reads: 'Carry Under License (Courseware) has a non-transferable'. The Oracle logo is at the bottom right.

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

Configuring SSL for an Oracle WebLogic Server

You configure SSL through the Administration Console:

1. Navigate to the server instance and click Lock & Edit.
2. Click the Configuration > SSL tab.
3. Enter the Keystore information and click Save.

Identity and Trust Locations: Indicates where SSL should find the server's identity (certificate and private key) as well as the server's trust (trusted CAs). If set to Keystores, SSL retrieves the identity and trust from the server's key store (that is configured on the server). The "Files or keystore providers" option is meant for use with older versions of WLS and is deprecated.

For a more secure deployment, Oracle recommends saving private keys in a keystore.

Road Map

- WLS Security Architecture overview
- Users and groups
- Protecting application resources
- Protecting communications
- Protecting against attacks
 - Types of attacks
 - Protecting against man-in-the-middle attacks
 - Protecting against denial of service (DoS) attacks
 - Protecting against large buffer attacks
 - Protecting against connection starvation



ORACLE

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

Protecting Against Attacks

WLS can help protect applications against several attacks:

- Man-in-the-middle attacks
- DoS attacks
- Large buffer attacks
- Connection starvation attacks



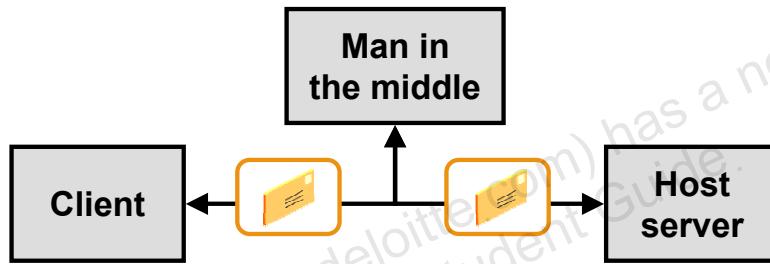
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Protecting Against Attacks

In the following pages, attacks and countermeasures are described in detail.

Man-in-the-Middle Attacks

- In the “man-in-the-middle” attack, a third party poses as a destination host intercepting messages between the client and the real host.
- Instead of issuing the real destination host’s SSL certificate, the attacker issues his or her own hoping that the client would accept it as being from the real destination host.



ORACLE

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

Man-in-the-Middle Attacks

When you use SSL, servers that do not use a certificate signed by a trusted CA are vulnerable to the “man-in-the-middle” attacks.

If a client accepts the attacker’s certificate, the “man-in-the-middle” can decrypt and forward the traffic to and from the real destination host and monitor it.

Man-in-the-Middle: Countermeasures

- The “man-in-the-middle” attacks can be resisted by using a Hostname Verifier.
- A Hostname Verifier validates that the host to which an SSL connection is made is the intended or authorized party.
- WLS provides a Hostname Verifier by default.
- A custom Hostname Verifier can be created by implementing the `weblogic.security.SSL.HostnameVerifier` interface.



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

Man-in-the-Middle: Countermeasures

A Hostname Verifier is useful when an Oracle WebLogic Server or a WebLogic client acts as an SSL client to another application server. It prevents the “man-in-the-middle” attacks.

By default, Oracle WebLogic Server, as a function of SSL handshake, compares the common name in `SubjectDN` of the SSL server’s digital certificate with the host name of the SSL server that is used to initiate the SSL connection. If these names do not match, the SSL connection is dropped. The dropping of the SSL connection is caused by the SSL client, which validates the host name of the server against the digital certificate of the server.

If anything but the default behavior is desired, you can either turn off host name verification or register a custom Hostname Verifier. Turning off host name verification leaves Oracle WebLogic Server vulnerable to the “man-in-the-middle” attacks.

Note: Turn off host name verification when you use the demo digital certificates that are shipped with Oracle WebLogic Server. You can turn off host name verification in the following ways:

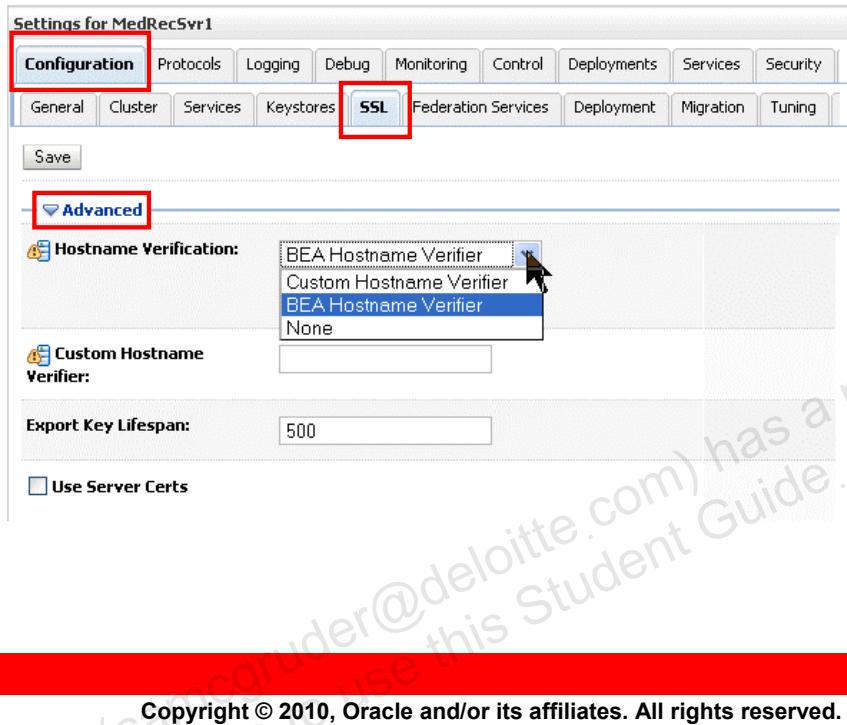
- In the Administration Console, select the Hostname Verification Ignored attribute under the SSL tab on the Server node.
- On the command line of the SSL client, enter the following argument:
-D`weblogic.security.SSL.ignoreHostnameVerification=true`

Man-in-the-Middle: Countermeasures (continued)

- To use a custom Hostname Verifier, create a class that implements the `weblogic.security.SSL.HostnameVerifier` interface and define the methods that capture information about the server's security identity.
 - In the Administration Console, define the class for your Hostname Verifier in the Hostname Verifier attribute (an Advanced option under the Configuration > SSL tab for the server).
 - On the command line, enter the following argument:
`-Dweblogic.security.SSL.HostnameVerifier=hostnameVerifier`
where `hostnameVerifier` is the name of the class that implements the custom Hostname Verifier.

Configuring a Hostname Verifier

The hostname verifier is enabled by default.



ORACLE

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

Configuring a Hostname Verifier

To configure a custom Hostname Verifier, perform the following steps:

1. If you have not already done so, in the Change Center of the Administration Console, click Lock & Edit.
2. In the left pane of the Console, expand Environment and select Servers.
3. Click the name of the server for which you want to configure a Hostname Verifier.
4. Select Configuration > SSL and click Advanced at the bottom of the page.
5. Select the appropriate Hostname Verifier in Hostname Verification.
6. Enter the name of the implementation of the `weblogic.security.SSL.HostnameVerifier` interface in the Custom Hostname Verifier field.
7. Click Save.
8. To activate these changes, in Change Center of the Administration Console, click Activate Changes.

Note: Not all changes take effect immediately; those marked with a triangle exclamation require a restart of the server.

Denial of Service Attacks

- DoS attacks are attempts by attackers to prevent legitimate users of a service from using that service.
- There are three basic types of attack:
 - Consumption of scarce, limited, or nonrenewable resources
 - Destruction or alteration of configuration information
 - Physical destruction or alteration of network components



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

Denial of Service Attacks

DoS attacks can disable your computer or your network. Depending on the nature of your enterprise, this can effectively disable your organization.

Some DoS attacks can be executed with limited resources against a large, sophisticated site. This type of attack is sometimes called an “asymmetric attack.” For example, an attacker with an old PC and a slow modem may be able to disable much faster and more sophisticated machines or networks.

Examples include attempts to:

- “Flood” a network, thereby preventing legitimate network traffic
- Disrupt connections between two machines, thereby preventing access to a service
- Prevent a particular individual from accessing a service
- Disrupt service to a specific system or person

Denial of Service Attacks: Countermeasures

Harden WLS against DoS attacks by:

- Filtering incoming network connections
- Configuring consumable WLS resources with the appropriate threshold and quotas
- Limiting access to configuration information and configuration tools
- Limiting access to back up configuration files
- Preventing unauthorized access by protecting passwords against password-guessing attacks



ORACLE

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

Denial of Service Attacks: Countermeasures

You can also use tools such as Oracle Adaptive Access Manager (OAAM) that can effectively prevent unauthorized accesses.

Filtering Network Connections

- WLS can be configured to accept or deny network connections based on the origin of the client.
- This feature can be used to restrict the:
 - Location from which connections to WLS are made
 - Type of connection made, for example:
 - Allow only SSL connections and reject all others
 - Deny all t3 protocol connections



ORACLE

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

Filtering Network Connections

To configure connection filtering in the server, create a `ConnectionFilterImpl` class that implements the `weblogic.security.net.ConnectionFilter` interface (minimum requirement) and the `ConnectionFilterRulesListener` interface (optional). Use the Administration Console to install the class in Oracle WebLogic Server so that the server examines requests as they occur, and then accepts or denies them.

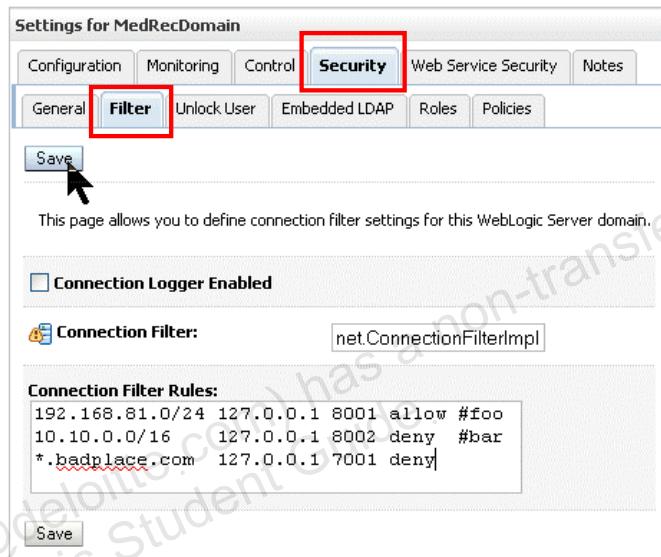
When a Java client or a Web browser client tries to connect to Oracle WebLogic Server, Oracle WebLogic Server constructs a `ConnectionEvent` object and passes it to the `accept()` method of your connection filter class. The `ConnectionEvent` object includes the remote IP address (in the form of `java.net.InetAddress`), the remote port number, the port number of the local Oracle WebLogic Server, and a string specifying the protocol (http, https, t3, t3s, or IIOP).

To filter network connections, either create a class that implements the `ConnectionFilter` interface and install it using the Administration Console, or use the default filter. The default network filter is always on. You just have to configure the rules. There is no need to implement a class for the default filter.

The connection filter class (`ConnectionFilterImpl`) examines the `ConnectionEvent` object and either accepts the connection by returning or denies the connection by throwing a `FilterException`.

Connection Filter

- Freeform Filter Rules Format:
 - targetAddr localAddr localPort action protocols
- Comments start with #
- Address mask can be:
 - Bits, e.g. /16
 - /255.255.0.0
- Action can be:
 - allow
 - deny
- Protocols can be:
 - http, https
 - t3, t3s
 - ldap, ldaps
 - iiop, iiops
 - com



ORACLE

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

Connection Filter

Using the Administration Console, access the domain (top) node in the navigation panel.

1. Click the Security > Filter tab.
2. In the Connection Filter field, specify the connection filter class to be used in the domain.
 - To configure the default connection filter, specify `weblogic.security.net.ConnectionFilterImpl` (only part of the name is showing on the screen).
 - To configure a custom connection filter, specify the class that implements the network connection filter. This class must also be present in CLASSPATH for Oracle WebLogic Server.
3. Connection Filter Rules are written on single lines, white space is ignored. The rules are evaluated in order. When the first rule matches, evaluation stops (so order is important). You can put a catch-all rule at the bottom that says:
`0.0.0.0/0 * * deny`
in case any other rule is not satisfied.
4. After adding Filter Rules, click Save.

Excessive Resource Consumption

- Denial of service can come from consuming server-side resources used by Web applications:
 - Intentionally generating errors that will be logged, consuming disk space
 - Sending large messages, many messages, or delaying delivery of messages in an effort to cripple JMS
 - Disrupting network connectivity through “connection starvation”
 - Consuming system memory through “large buffer attacks”
- The effect of these attacks can be reduced by setting the appropriate quotas and threshold values.

ORACLE

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

Excessive Resource Consumption

The Oracle WebLogic Server resources can be vulnerable to abuse. A malicious piece of code can consume all the available database connections or cripple a service such as JMS by sending many large messages or delaying the delivery of messages.

You can reduce the effect of these attacks by using the Administration Console to set reasonable quotas and threshold values for each resource. You can also set the size of the log files and their rotation values to limit the amount of disk space that is consumed.

Large Buffer Attacks

- Individuals can try and bring down a Web site by sending a large buffer of data, which starves the system of memory.
- Administrators can combat this attack by setting a threshold for incoming data.



ORACLE

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

Large Buffer Attacks

Hackers try to bring down a Web site in a variety of ways. One particular way is referred to as large buffer attacks because hackers send large buffers of data to the server that starves the server of memory. Oracle WebLogic Server allows administrators to set a limit to the amount of HTTP data that can be posted to their servers. Administrators can use the Administration Console to manage this threshold. Any requests that exceed this threshold are denied access to the server.

Setting the Post Size

The screenshot shows the 'Settings for MedRecSvr1' interface. In the left pane, under the 'Protocols' tab, the 'HTTP' tab is selected. The 'Max Post Size' field is set to 512 and is highlighted with a red box. A callout bubble with the text '-1 (default) means unlimited size.' points to this field. In the right pane, the 'HTTP Max Message Size' field is also highlighted with a red box and has a value of -1. Other configuration options like 'Post Timeout', 'Duration', and 'HTTPS Duration' are also visible.

ORACLE

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

Setting the Post Size

The Max Post Size parameter determines the size of a data buffer that a server allows for reading HTTP POST data in a servlet request. A value less than 0 (such as -1) indicates an unlimited size.

To set the threshold of the request sizes that can be posted to the server, perform the following steps:

1. In the left pane, select the server that you want to set the limit on.
2. Click the Protocols > HTTP tab in the right pane.
3. Set Max Post Size. This is the threshold amount for the incoming requests. In this example, the maximum amount of data sent is 512 KB.
4. After you have finished entering your information, click Save to save your changes.

Similarly, HTTP Max Message Size limits the number of bytes allowed in messages that are received over the HTTP protocol. If you configure custom network channels for this server, each channel can override this maximum message size. This maximum message size helps guard against a DoS attack in which a caller attempts to force the server to allocate more memory than is available, thereby keeping the server from responding quickly to other requests.

Note: You need to restart the server after making these modifications.

Connection Starvation

- Individuals can try and take down a Web site by sending small, incomplete messages that cause the server to wait.
- Administrators can combat this attack by setting a threshold.
- Connections time out while waiting for the remainder of the data if they have reached the threshold set by the administrator.



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

Connection Starvation

Another way that individuals can try and harm a Web site is by sending small, incomplete messages to the server. The server then waits for the completion of the message, in effect unduly burdening the server. Oracle WebLogic Server enables administrators to set a threshold for the time Oracle WebLogic Server will wait for the completion of the message. The administrator sets the time-out feature in the Administration Console and any connections that are still waiting for the completion of the message longer than this limit are canceled.

Connection Starvation

HTTPS Duration:	60
Frontend Host:	
Frontend HTTP Port:	0
Frontend HTTPS Port:	0
<input type="checkbox"/> WAP Enabled	

ORACLE

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

Connection Starvation (continued)

To set the threshold of Post Timeout and Max Post Time, perform the following steps:

1. In the left pane, select the server that you want to set the limit on.
2. In the right pane, click the Protocols tab.
3. Click the HTTP tab.
4. Set Post Timeout, which is the maximum amount of time that Oracle WebLogic Server waits for the next packet.
5. After you have finished entering your information, click Apply to save your changes.

Note: You need to restart the server after making these modifications.

Similarly, you can also set the amount of time this server waits before closing an inactive HTTPS connection by using the HTTPS Duration parameter. The value you specify is in seconds. The default of 60 seconds may be very large for some applications.

You specify the number of seconds during which to keep the HTTPS active before timing out the request.

User Lockout

- Individuals attempt to hack into a computer using various combinations of usernames and passwords.
- Administrators can protect against this security attack by setting the lockout attributes.
- The administrator can unlock a locked user using the console.

```
:  
<Mar 2, 2010 2:42:36 PM EST> <Notice> <Security> <BEA-090078> <User johndoe  
in security realm myrealm has had 5 invalid login attempts, locking account  
for 30 minutes.>
```



ORACLE

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

User Lockout

Password guessing is a common type of security attack. In this type of attack, a hacker attempts to log in to a computer by using various combinations of usernames and passwords. Oracle WebLogic Server provides a set of attributes to protect passwords and user accounts in a security realm.

Configuring User Lockout

Settings for myrealm

Configuration Users and Groups Roles and Policies Credential

General RDBMS Security Store **User Lockout** Performance

Save

Lockout Enabled

Lockout Threshold:

Lockout Duration:

Lockout Reset Duration:

Lockout Cache Size:

Lockout GC Threshold:



ORACLE

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

Configuring User Lockout

The User Lockout feature enables you to prevent attack from hackers using a compromised user account. The User Lockout attributes apply to the security realm and all its security providers. If you are using an authentication provider that has its own mechanism for protecting user accounts, disable the Lockout Enabled attribute.

- **Lockout Threshold:** The maximum number of consecutive invalid login attempts before the account is locked out. For example, with the setting of 1, the user is locked out on the second consecutive invalid login. Minimum is 1 and the default is 5.
- **Lockout Duration:** The number of minutes that a user account is locked out. Minimum is 0 and the default is 30.
- **Lockout Reset Duration:** The number of minutes within which consecutive invalid login attempts cause the user account to be locked out. Minimum is 1 and the default is 5.
- **Lockout Cache Size:** The number of invalid login records that the server places in a cache. The server creates one record for each invalid login. Minimum is 0 and the default is 5.
- **Lockout GC Threshold:** The maximum number of invalid login records that the server keeps in memory. If the number of invalid login records is equal to or greater than this value, the server's garbage collection purges the records that have expired.

If a user lockout security event occurs on one node of a cluster, the other nodes in the cluster are notified of the event and the user account is locked on all the nodes in the cluster. This prevents a hacker from systematically breaking into all the nodes in a cluster.

Unlocking Users

Settings for MedRecDomain

Configuration Monitoring Control **Security** Web Service Security Notes

General Filter **Unlock User** Embedded LDAP Roles Policies

Save

If a user unsuccessfully attempts to log into a WebLogic Server server more than the configured number of retry attempts, then they are locked out of further access.

This page allows you to unlock a locked user so that they can log in again.

Unlock User:

Save



ORACLE

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

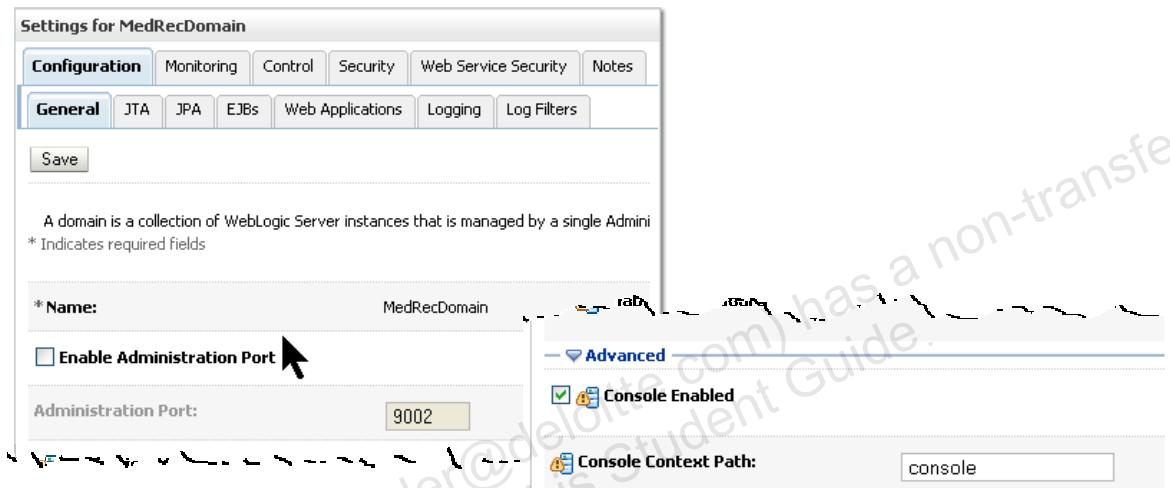
Unlocking Users

If a user unsuccessfully attempts to log in to a WebLogic Server more than the configured number of retry attempts, they are locked out of further access. The Unlock User page allows you to unlock a locked user so that they can log in again.

Note: If a user account becomes locked and you delete the user account and add another user account with the same name and password, the User Lockout attribute will not be reset—that is, the added user may remain in the lockout status.

Protecting the Administration Console

- You can configure a separate administration port for all administration traffic.
- You can change the context path of the console.
- You can disable the console (application).



Protecting the Administration Console

By configuring a separate administration port for administration tasks, you do not expose the administration ports to other application ports. Before you enable an administration port, you ensure that all the servers in the domain are configured with SSL.

Similarly, you can reconfigure the context path of the console so that it does not remain the generally known /console.

Finally, in a production environments where you are less likely to make configuration changes regularly, you can disable the console application.

Quiz

The Hostname Verifier is one measure for combating this type of attack:

- a. Large buffer
- b. Connection starvation
- c. Man in the middle
- d. User lockout

ORACLE

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

Quiz

To counter connection starvation attacks, you can set:

- a. Max Post Size
- b. Post Timeout
- c. Hostname Verifier
- d. User lockout



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

Summary

In this lesson, you should have learned how to:

- Describe the process of configuring SSL
- Use the `keytool` utility to configure keys and obtain digital certificates
- Configure SSL for the WLS server
- Configure countermeasures for some Web-based attacks such as:
 - Man in the middle
 - Denial of service
 - Large buffer
 - Connection starvation

ORACLE

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

Practice 20: Overview Configuring Keystores

This practice covers the following topics:

- Using keytool to generate an identity keystore that contains a private key and a self-signed public certificate
- Configuring keystores using the Administration Console
- Configuring SSL for a managed server

ORACLE

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

21

Backup and Recovery Operations

ORACLE®

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

Objectives

After completing this lesson, you should be able to:

- Recommend a backup and recovery strategy
- Perform a full offline backup and recovery
- Perform an online and offline domain backup
- Perform an offline domain recovery
- Perform an Instance Home backup and recovery



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

Objectives

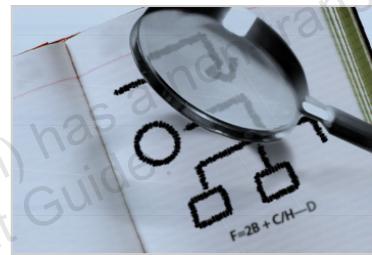
Scenario

As the middleware administrator, you need to plan a reasonable backup strategy that balances risk against inconvenience. Backing up once a month is too infrequent, whereas once an hour is too frequent, so what is the right balance? Given that you will do far more backups than recoveries, a plan that favors backup by shortening the time to create the backups at the expense of lengthening and complicating the recovery might be worth trying. Given that different backup strategies cause different kinds of recoveries, you plan to time how long it takes to do a recovery to help create service-level agreements (SLA).

Note the distinction between *restore* and *recover*: restore is a pure file system copy operation, whereas recovery is restore plus some extra operations to get to a specific point-in-time-recovery.

Road Map

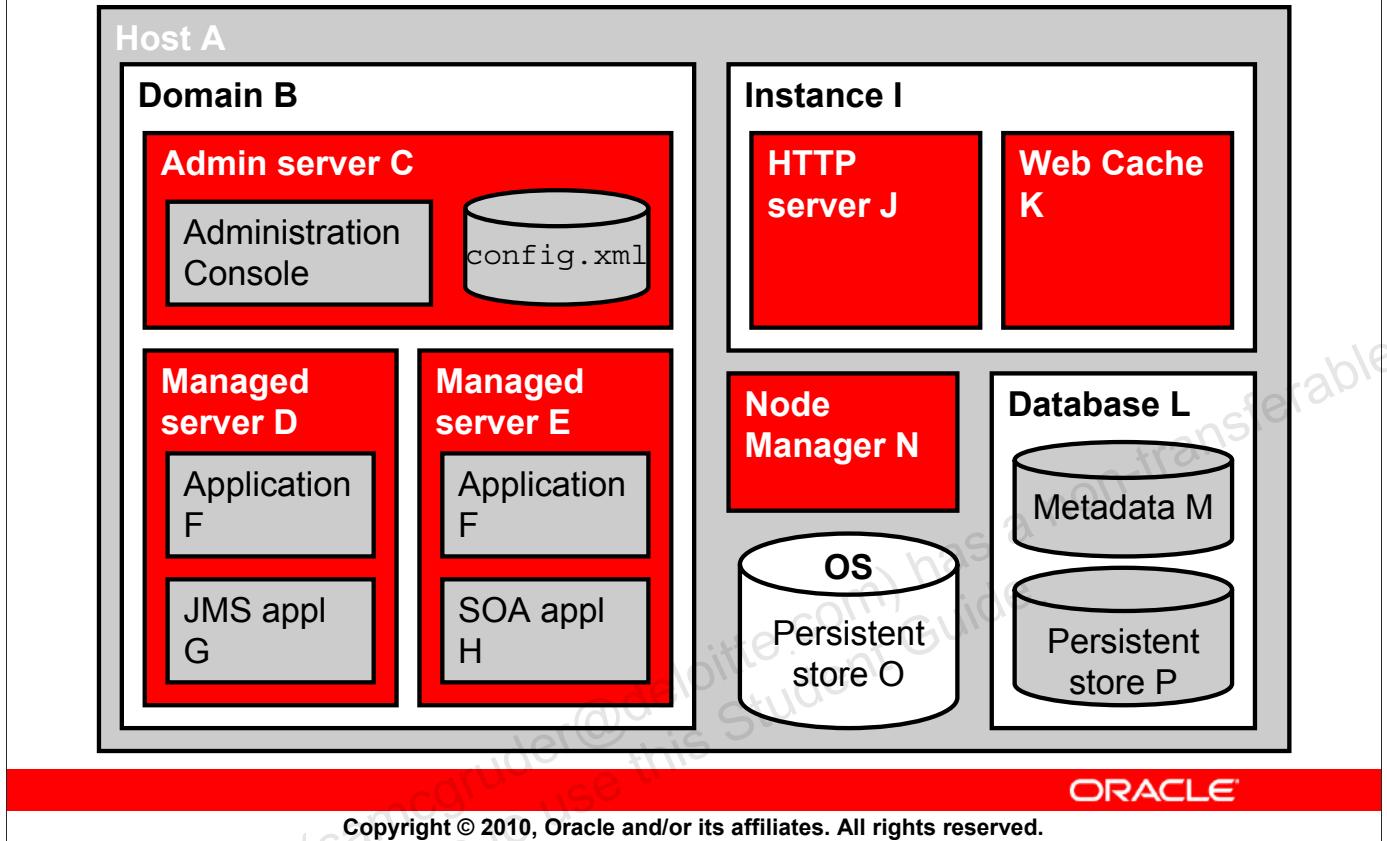
- Backup
 - Full
 - Incremental
 - Online
 - Offline
- Recovery



ORACLE

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

Review of Terms and Components



Review of Terms and Components

- Host:** The computer may have redundant CPUs, RAID disks, and/or other hardware failover features.
- Domain:** WebLogic Server is an example of a system component domain, and Oracle HTTP Server and Oracle Web Cache are system component domains as well. In this case, a WebLogic Server domain consists of at least one administration server and zero or more managed servers. These servers are Java components.
- Administration server:** This server is required at the initial start of a managed server, but not thereafter. The administration server contains the master `config.xml` file, which is copied to managed servers at various times (startup, configuration changes, and so on).
- Managed server:** This server runs Java EE applications. The server may be part of a cluster.
- Managed server:** This server is the SOA server. SOA requires a metadata repository on a database (created by the Repository Creation Utility [RCU]) or in a plain file.
- Application:** An application may be deployed on a cluster or on several stand-alone servers.
- JMS application:** Messages can be stored either in a database persistent store or in a plain file persistent store.
- SOA application:** SOA requires a metadata repository, either in a database or a plain file.

Review of Terms and Components (continued)

- I. **Instance:** This is similar to a domain, but it contains system components. A system component is a non-Java component managed by the Oracle Process Manager and Notification (OPMN) server.
- J. **Oracle HTTP Server:** Based on the Apache 2.2.10 infrastructure, this includes modules developed specifically by Oracle. The features of single sign-on, clustered deployment, and high availability enhance the operation of Oracle HTTP Server.
- K. **Web Cache:** Because this is a *cache*, there is no permanent data to back up or recover, but there are configuration files and logs. Because there is no live data that you care about, the backup can be performed online. There is no need to worry about consistency or run-time artifacts.
- L. **Database:** Assuming this is an Oracle database, the backup tool is Recovery Manager (RMAN), capable of performing online backups and automated recovery to either any point in time or a complete recovery. A Flashback log (if configured) also provides a rolling recovery window. If the environment permits offline backup, after the processes are all stopped, a simple OS copy of all files will work. RMAN tasks are typically performed by the DBA and are outside the scope of this course.
- M. **Metadata:** Required for SOA, this is created by RCU. Use the database tools for backup and recovery.
- N. **Node Manager:** One per host, the Node Manager can autorestart managed servers that fail.
- O. **Persistent store:** This is an OS file that can contain JMS transactions.
- P. **Persistent store:** This is a database schema that can contain JMS transactions. Use the database tools for backup and recovery.

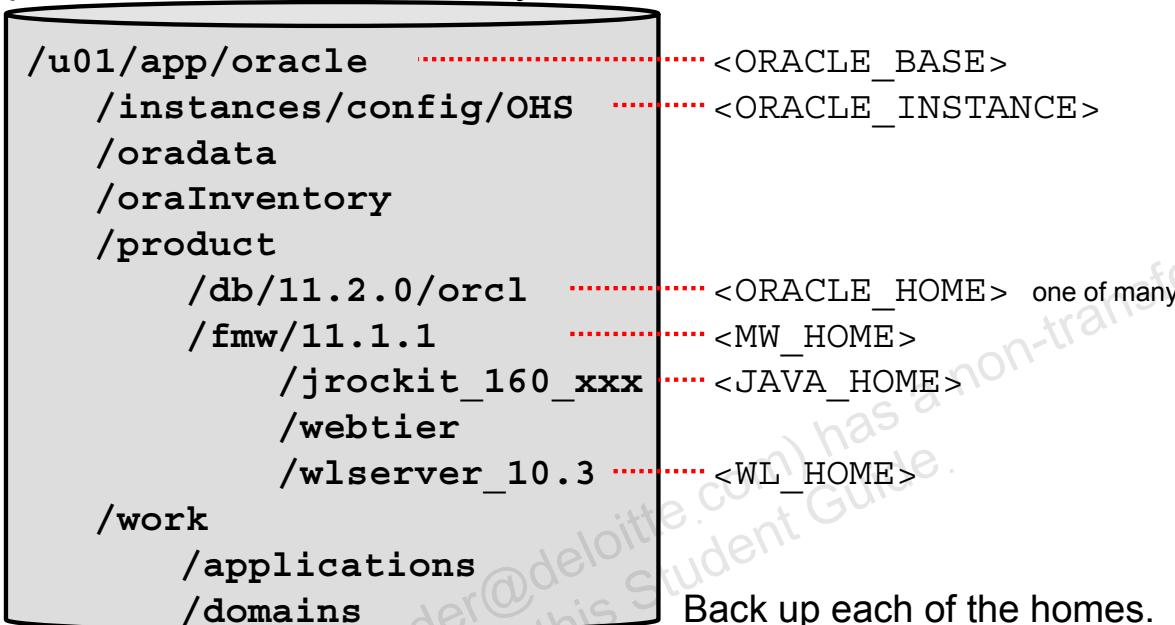
Static artifacts: The program binaries do not change very often. Their backup schedule might be only after patches, monthly, or even longer.

Run-time artifacts: These objects change frequently, even multiple times per second in the case of logs. Configuration objects may change several times per day, though typically they remain unchanged for long periods of time.

Persistent stores: These objects may change very frequently, even hundreds of times per second, depending on the volume of data traffic. A high-performance solution may be required so as to not lose data.

Homes: Oracle, Middleware, WebLogic

You can set up the disk in any way you like; this is only a portion of one suggested layout:



Back up each of the homes.



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

Homes: Oracle, Middleware, WebLogic

This is the layout of the disks in the lab. Each of the homes can be a point for starting an incremental backup. Starting from <ORACLE_BASE> would be a full offline backup.

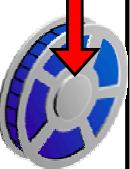
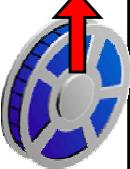
Oracle home: There can be several simultaneous <ORACLE_HOME>s. An Oracle home contains installed files necessary to host a specific product. Shown is the home for the database. Not shown might be a <SOA_ORACLE_HOME> and a <WC_ORACLE_HOME> (Web Cache). For example, the SOA Oracle home contains a directory that contains binary and library files for Oracle SOA Suite. The WebLogic Server home also consists of its installed files.

An Oracle home resides within the directory structure of the Middleware home. Each Oracle home can be associated with multiple system component domains or Oracle WebLogic Server domains. The WebLogic Server Home directory is a peer of Oracle Home directories. In order to keep all the multiple Oracle homes from conflicting with each other, they should be defined only in the scripts that start a particular process, not globally defined in a .profile nor using the source command.

Middleware home: The Middleware home consists of the Oracle WebLogic Server home and one or more Oracle homes, such as SOA home and Web Cache home.

Instance home: The Instance would contain the Oracle HTTP Server and Web Cache configuration files.

Understanding Backup and Recovery

<u>Backup</u>	<u>Recovery</u>
<ul style="list-style-type: none"> • Scheduled • At least weekly (to capture logs) • Different tools for different components 	 <ul style="list-style-type: none"> • Unscheduled (usually) • At least annually (if only to test procedures) • Not necessarily the reverse of backup, may be new tools 

- Protects against failures of hardware, software, power, environmental disasters, accidental and malicious changes, and more
- Guarantees a point of recovery, minimizes loss of business availability, insures an SLA, may satisfy legal requirements
- May impact business
- May be hardware and software

ORACLE

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

Understanding Backup and Recovery

Commonly, the terms “backup” and “recovery” imply the use of secondary media to copy some data for later use. That kind of backup and recovery involves an offline or cold storage of the data such that if an outage occurs, then some process (human or automated) requires some time to get the system back up and running. Alternatively, “redundancy” and “failover” are additional means by which to back up and recover the data in more of an online or warm or hot storage mode, thus reducing, or even eliminating the switchover time. If an outage occurs with redundancy and failover implemented, it is often undetectable by the user. The following are different forms of backup and recovery:

- Redundant disks in a SCSI array
- Multiple servers configured on multiple machines in a cluster with an application deployed on the cluster
- The ability to cancel all pending changes to a configuration
- The architecture of the Oracle 11g Database with inherent transaction logging

In addition to those very significant features, a media backup plan is essential. The most common problem that requires a backup and recovery is when a person who is authorized to make changes accidentally commits a wrong change. Usually, the mistake is realized within seconds and all that is needed is a mechanism that will enable the user to go back to a very recent version of the configuration. A more serious problem is when there is a complete loss of

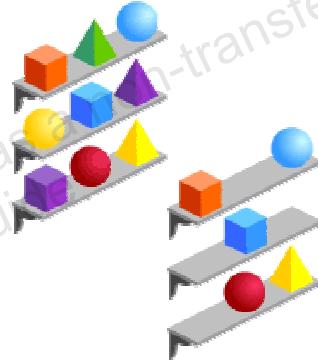
Understanding Backup and Recovery (continued)

the computer hosting the WebLogic component. There is no single point of failure in the Fusion Middleware architecture, but there may be an impact in service (for example, no configuration changes can be made while the administration server is down).

Backup and recovery policies may impact your business both financially and in terms of availability (required maintenance windows).

Types of Backups

- Online
 - Nondisruptive
 - Possibly inconsistent
 - Can be tricky, especially for database
- Offline
 - Requires *all* processes to be stopped
 - Very easy
- Full
 - Easier to recover
 - Slower to create
- Incremental
 - Harder to recover
 - Faster to create



ORACLE

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

Types of Backups

Online

If your environment requires 24×7 availability, you have no choice but to perform an online backup. Different components require different tools to perform online (also known as hot or inconsistent) backups. Inconsistent is not bad in itself; it just means that if the backup takes an hour to complete and you start at 1:00 AM, the files at 1:02 AM will be in a different state than those backed up at 1:59 AM. To accommodate this, there needs to be some kind of online transaction log recording the changes occurring from 1:00 AM until 2:00 AM. This log needs to be incorporated into the recovery, and the logs themselves get backed up at a different time (usually, after they rotate).

Offline

If you can afford to shut down the entire middleware tier (application servers, database, Web servers, and so on) for maintenance during some regularly scheduled time, an offline (also known as cold or consistent) backup is very simple. Using OS tools such as TAR or ZIP, the backup is guaranteed to be consistent. Make sure you preserve file permissions on UNIX systems.

Types of Backups (continued)

Full

After the initial installation, or after a major set of patches, a full backup should be performed. Often, this is done before going live, so the system is offline. It is very difficult (if not impossible) to perform a full backup online. If there is a complete loss of a host (for example, a disaster such as a fire or flood), recovery is simple; just copy the backup files to the new bare-metal host and boot. Name the backup so as to include the date—for example, `my_full_backup_2010_03_30.tar`—and keep several generations of them in case you accidentally capture “the problem” in the most recent backup.

Incremental

Considering that the executable files and the configuration files are usually backed up separately, most backups are partially incremental. Backing up only changes may require several sets of backups recovered in order to perform a full recovery. RMAN can help automate this for databases, especially if the backups are kept online (on disk as opposed to tape).

You can make an incremental backup at the functional level. For example, you can make a WebLogic backup from `<WL_HOME>`, make an instance backup from `<ORACLE_INSTANCE>`, make a database backup from `<ORACLE_HOME>`, and so on. Within WebLogic, make a backup of all domains and then make backups of individual domains. The disadvantage of doing this is that the backup process will take a long time, but the advantage is that the recovery process can be greatly simplified. Alternatively, if you do not make so many different kinds of incremental backups, the backup procedure will complete faster, but now you have complicated and lengthened your potential recovery time. It is a delicate tradeoff balancing storage space versus time versus complexity.

Backup Recommendations

- After the initial domain is created (offline)
- Scheduled backups (online)
- After the component or the cluster changes (online)
- Before application deployment (online)
- Before patches or upgrades (offline)
- Database backups (online) for:
 - LDAP
 - Persistent stores
 - SOA repository



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

Backup Recommendations

The initial software installation and most patches and upgrades require the servers to be offline anyway, so before and after the patches and upgrades is a good time to perform backups.

Many of the online configuration backups can be automatic by enabling configuration archive (discussed in the following slides).

The database should be in archivelog mode and then backed up with RMAN. In addition, the database should be configured with redundant critical files (for example, control files) and multiplexed critical logs (for example, redo logs). As an added high availability measure, the database can be made completely redundant by using RAC.

Limitations and Restrictions for Backing Up Data

- You should not be adding users or changing permissions while backing up the Lightweight Directory Access Protocol (LDAP).
- Online persistent stores by nature are going to be an inconsistent backup.
 - Database backups can accommodate inconsistencies.
 - File-based stores and OS copies cannot accommodate online backup.
- HTTP session states and cookies information may be lost.
 - In-memory replication may lose the state.
 - JDBC replication of the HTTP session state solves this problem.



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

Limitations and Restrictions for Backing Up Data

None of these restrictions apply to offline backups; they apply only to online backups. In many cases, the WebLogic Server has the option to be configured to use either database or file storage for information. Choosing database is always a safer option, but you pay for it with complexity and perhaps a speed penalty. If you have a database anyway, and the DBA is backing up the database anyway, some additional WebLogic Server files should not be any additional effort, so it is worth the security to specify database storage when possible over OS file storage. For files such as configuration XML; application JARs, WARs, or EARs; and properties files; database storage is not an option.

Performing a Full Offline Backup

1. Shut down all processes:
 - Stop WebLogic via the Administration Console.
 - Shut down the database.
 - Stop the Listener and the Node Managers.
 - Stop the Enterprise Manager and the emAgent.
 - Stop Web Cache and HTTP server via OPMN.
2. Perform the backup via OS tools:
 - If using TAR, make sure that you keep permissions.
 - If using ZIP, make sure that you include empty directories.
3. Test the backup by performing recovery on another computer:
 - Ideally, use an alternate computer in an alternate data center.
 - Time the recovery for SLA input.
4. Store the backup offsite.



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

Performing a Full Offline Backup

Shutting Down

Stop all deployed applications so that you can shut down all servers. Verify the Node Managers and emAgent PIDs. Stop the Node Managers and emAgents. In SQL*Plus (`sqlplus / as sysdba`), issue `shutdown immediate`. This may take a while (despite the name, it is not “immediate”). Stop the database listeners and the Enterprise Manager console. Stop all OPMN-managed utilities (for example, OHS and Web Cache) using `opmnctl stopall`.

Performing Backup

In the lab, all the product and configuration information is stored in `/u01/app/work`. Signed on as root, from the `root` directory, use the appropriate operating system backup utilities (for example, `tar` or `winzip`):

```
tar -zcvpf mybackup1.tar /u01/app/work /etc/ora* etc/hosts
```

There may be more sophisticated options to exclude `/tmp/` files and to include parts of other applications, but this will do as a start. The sequence number 1, 2, 3, might be replaced with the `date_time` in the name of the TAR file. If the directories are backed up from the root, you do not need to worry about where to recover them to; that information will be part of the backup.

Performing a Full Backup (continued)

Testing Restore

Make sure that the first backup you took is successful. You need to do this very early in the life of the system while there is no urgent need. Many administrators will tell you unfortunate stories of how they found out only during an emergency that several tapes that they dutifully preserved were blank for some simple overlooked reason. Later, you need to perform scheduled recoveries to make sure that the processes are all still working.

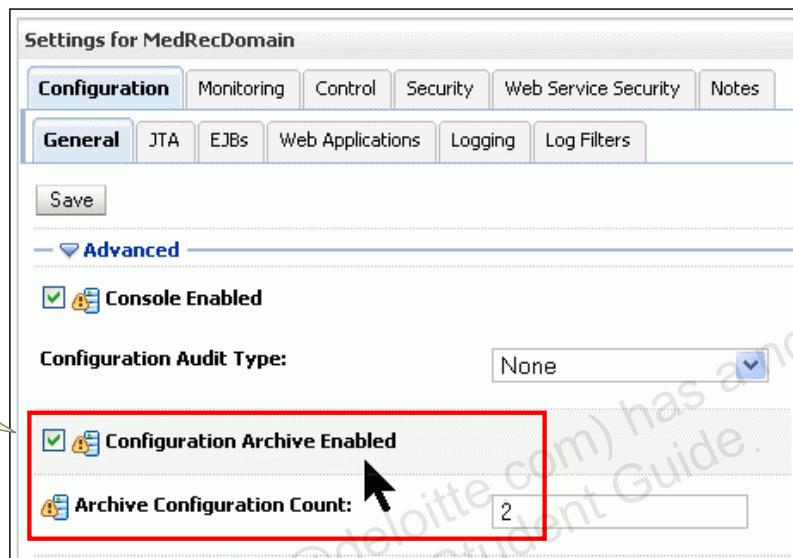
From the `root` directory, signed on as the `root` user, enter:

```
tar -zxvpf mybackup1.tar
```

Because you have signed on as `root`, it is vital to make sure that the `-p` switch is used in the `tar` command to preserve the original owners and group permissions (for example, `oracle` and `oinstall` versus `root`). Restart all processes to test the recovery and make sure that it is complete.

Backing Up a Domain Configuration

- Enable autobackup of configuration.
- Check new JAR files and directories.



ORACLE

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

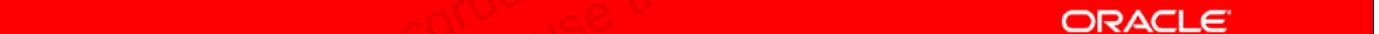
Backing Up a Domain Configuration

In Domain > Configuration > General > Advanced, you can enable autobackup at the domain level. Each startup of the administration server creates two files: config-booted.jar and config-original.jar in the domain directory. In addition, each saved change of the configuration file makes a backup named configArchive/config-n.jar, where n is a sequential number. Archive Configuration Count limits the number of retained configuration JARs, so that in the example shown, there are never more than two kept: the most recent backup and the one immediately before that. Older backups are automatically deleted. If you made a series of mistakes, this provides a very easy way to return to a previous recent configuration. However, be aware that a typical configuration change requires clicking Activate Changes a few times, and each one then cycles the stored JARs. You may want a higher number such as 10 or 20 for the count. An example from the MedRecDomain directory:

```
[oracle@edvmr1]# cd /u01/app/work/domains/MedRecDomain
[oracle@edvmr1]# ll conf*
drwxr-x--- 11 oracle oinstall 4096 Mar 23 16:51 config
drwxr----- 2 oracle oinstall 4096 Mar 25 08:58 configArchive
-rw-r----- 1 oracle oinstall 12328 Mar 25 08:54 config-booted.jar
-rw-r----- 1 oracle oinstall 12328 Mar 25 08:54 config-original.jar
[oracle@edvmr1]# ll configArchive/
-rw-r----- 1 oracle oinstall 12339 Mar 25 08:59 config-2.jar
-rw-r----- 1 oracle oinstall 12328 Mar 25 09:03 config-3.jar
```

Backing Up an Instance Home

- Stop the Web tier (Oracle HTTP Server and Oracle Web Cache):
 - opmnctl stopall
 - opmnctl status
- Copy the Instance home:
 - As the superuser, change to the root directory.
 - Execute `tar -zcvpf myinstance1.tar $ORACLE_INSTANCE`.
- Restart all services:
 - opmnctl startall
 - opmnctl status

The red bar contains the word "ORACLE" in white capital letters.

ORACLE

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

Backing Up an Instance Home

There may be more sophisticated ways of not backing up the `/tmp/` files, but this is a good start. There is no facility for performing an online backup of the Instance home. After creating the backup, store a copy offsite. A sample Instance home might contain the following directories and files:

```
[oracle@edvmr1] $ ll instances
total 32
drwx----- 4 oracle oinstall 4096 Mar 26 1:38 auditlogs
drwx----- 2 oracle oinstall 4096 Mar 26 1:37 bin
drwx----- 5 oracle oinstall 4096 Mar 26 1:38 config
drwx----- 3 oracle oinstall 4096 Mar 26 1:37 diagnostics
drwx----- 3 oracle oinstall 4096 Mar 26 1:37 OHS
drwx----- 2 oracle oinstall 4096 Mar 26 1:37 tmp
drwx----- 3 oracle oinstall 4096 Mar 26 1:38 WebCache
-rw----- 1 oracle oinstall      9 Mar 26 1:38 webcacheAdmin1621.txt
```

Creating a Record of Installations

Create a record for your Oracle Fusion Middleware product installation. The record must contain:

- For each host:
 - Names and addresses
 - OS information
- For each installation:
 - Installation type, host, owner name and number, group name and number, environment profile and type of shell, directory structure, mount points, full path for Oracle home, and port numbers used by the installation

Store it offsite.



ORACLE

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

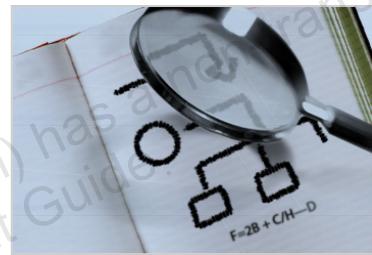
Creating a Record of Installations

You should maintain an up-to-date record of your Oracle Fusion Middleware installations in hard copy and in electronic form. You need this information in the event that you must restore and recover your installations to a new disk or host. The electronic form should be stored on a system that is completely separate from your Oracle Fusion Middleware that is being backed up. Your hardware and software configuration record should include:

- The following information for each host in your environment:
 - Host name, virtual host name (if any), domain name, IP address, hardware platform, and operating system release level and patch information
- The following information for each Oracle Application Server installation in your environment:
 - Installation type (for example, Infrastructure, or Java EE and Web Cache), host on which the installation resides, username, user ID number, group name, group ID number, environment profile, and type of shell for the operating system user that owns the Oracle home (/etc/passwd and /etc/group entries), directory structure, mount points, and full path for Oracle home, and port numbers used by the installation
 - For Oracle Database, the database version, patch level, base language, character set, global database name, and SID

Road Map

- Backup
- Recovery



ORACLE

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

Directories to Restore

- Binaries
 - Be mindful of preserving group ownership and permissions.
 - This should be read-only for most users.
- Configurations
 - If the last configuration *caused* the problem, recover to a point in time prior to that.
- Logs are:
 - Not required for recovery
 - Created if they do not exist
- Data
 - Database restores data within tablespaces, not directories.
 - RMAN *restore* brings data up to the last backup, then *recover* brings data up to a later point in time.

ORACLE

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

Directories to Restore

In most cases, recovery is performed offline. If you think that only one or two files are missing, you may be tempted to recover only those individual files from the system. But, instead, you should always recover whole directories because there may be other files that are related to these files.

If the directories were backed up from the root, you do not need to worry about where to recover them to. The full path information will be provided to the operating system because it is contained in the backup. Restore them as the `root` user, from the `root` directory, and they will go back to their correct hierarchies. Do not forget the `-p` switch in the `tar` or `jar` command to get the original owner and group information correct.

Recovery After Disaster

- Possible causes of failure:
 - Data loss
 - User error
 - Malicious attack
 - Corruption of data
 - Media failure
 - Application failure
- Recovery depends on the cause:
 - Repair
 - Replace
 - Relocate



ORACLE

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

Recovery After Disaster

If the problem was caused by a minor configuration error, the administrator may be able to reverse the steps and remove the problem without a formal recovery. If the problem requires replacing hardware, restoring full backups is a simple procedure. Recovery is complicated when you need to relocate some functions to an existing machine. According to the old configuration (and backups), the functions must be routed to the old name and address of A, but now according to the new configuration, the functions need to be routed to the new name and address of B.

Recovery of Homes

This applies to recovering a Middleware home, Oracle home, or Instance home after data loss or corruption:

1. Stop all processes.
2. Make a new full offline backup as a checkpoint (zip or tar -c).
3. Change directory to the affected home.
4. Use OS copy, tar -x, or unzip commands to restore the directories affected.
5. Make a new full offline backup (especially if you have been performing incremental backups up until this point).
6. Restart all processes.



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

Recovery of Homes

Make sure that all Fusion Middleware software is stopped so that this is an *offline* recovery. The most important rule in problem resolution is: “Do not make the problem worse.” By performing the two extra backups, you guarantee that you can at least put everything back to the way it was before you tried to help.

Assume that the last known good backup was sequence number 9. As an example, here is how to recover a damaged Instance home:

In the Administration Console, shut down all servers including the administration server:

```
opmnctl stopall
```

In SQL*Plus, shut down the database cleanly, that is, using “immediate”.

```
lsnrctl stop
tar -zcvpf mycheckpoint.tar /u01/app/work
tar -zxvpf myinstance09.tar
tar -zcvpf myfullbackup10.tar /u01/app/work
lsnrctl start
```

In SQL*Plus, start the database.

```
opmnctl startall
```

Start the administration server.

From the Administration Console, start the managed servers.

Recovery of a Managed Server

- If the software crashes, the Node Manager will automatically restart it.
- If the files are damaged, you can recover the files in their original places and restart the software.
- If the computer is damaged, perform either of the following:
 - Restore the files on a new host with the old computer name by using the following OS commands—for example, `copy`, `cp`, `tar`, or `unzip`.
 - Restore the files on another host with a different host name by using templates to extend the domain.

ORACLE

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

Recovery of a Managed Server

The original pack command that created the remote managed server can be used to re-create it in a recovery. The significant configuration and application files are stored at the administration server, so when the managed server comes back, it will first refresh all its configuration information and redeploy all its applications from the administration server.

Recovery of the Administration Server Configuration

Managed Server Independence (MSI) reduces the urgency to fix the outage.

Settings for MedRecSvr1

Configuration Protocols Logging Debug Monitoring Control Deployments Services Security

General Cluster Services Keystores SSL Federation Services Deployment Migration **Tuning**

Save

Use this page to tune the performance and functionality of this server.

Advanced

Managed Server Independence Enabled

Specifies whether this Managed Server can be started when the Administration Server is unavailable. [More Info...](#)

Period Length: The time interval in milliseconds of the heartbeat period. A value of 0 indicates that heartbeats are turned off. [More](#)

Idle Periods Until Timeout: The number of idle periods until peer is considered unreachable. [More Info...](#)

Save

ORACLE

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

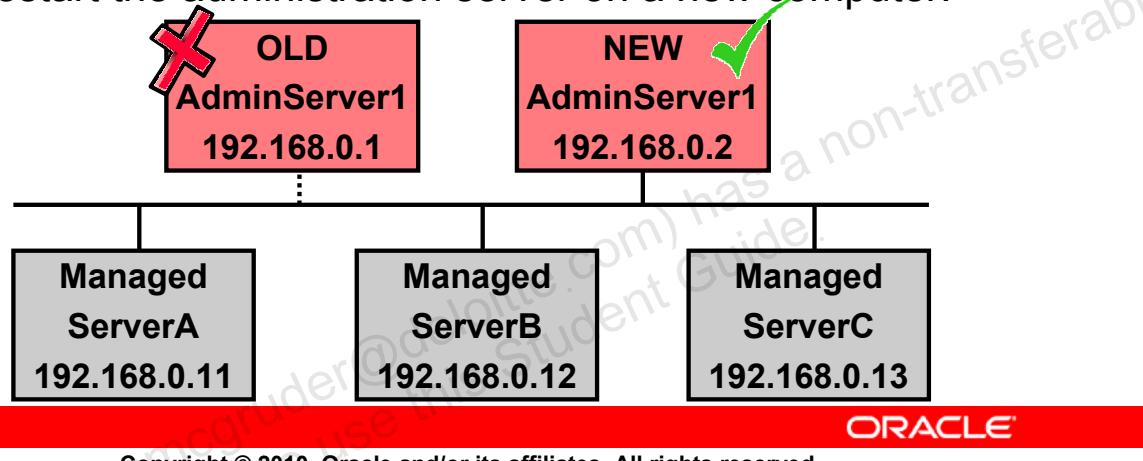
Recovery of the Administration Server Configuration

The administration server is required only for making changes to the active configuration; it is not required for the normal operation of the managed servers as long as the managed servers are in Managed Server Independence Enabled mode, which is the default. This allows you time to recover the administration server without any service outages. As shown in the screenshot, the heartbeat detected between the administration server and the managed servers is, by default, a one-minute period. After four minutes of not hearing from the administration server, the managed servers become independent. After the administration server is fixed, the heartbeats start up again and the managed servers deactivate their independence, but MSI is still enabled for a future event. These times can all be changed to suit your particular environment.

Restarting an Administration Server on a New Computer

Oracle WebLogic Server allows the creation of a backup of the administration server as follows:

1. Install Oracle WebLogic Server on a backup computer.
2. Copy the application files to a backup computer.
3. Copy the configuration files to a backup computer.
4. Restart the administration server on a new computer.



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

Restarting an Administration Server on a New Computer

If a hardware crash prevents you from restarting the administration server on the same computer, you can recover the management of the running managed servers as follows:

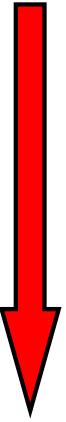
1. Install the Oracle WebLogic Server software on the new computer designated as the replacement administration server.
2. Make your application files available to the new administration server by copying them from backups or by using a shared disk. Your files must be available in the same relative location on the new file system as on the file system of the original administration server.
3. Make your configuration and security files available to the new administration computer by copying them from backups or by using a shared disk. These files are located under the directory of the domain being managed by the administration server.
4. Restart the administration server on the new computer.

When the administration server starts, it communicates with the already running managed servers via a Node Manager and informs the servers that the administration server is now running on a different IP address.

Note: You cannot have two administration servers at the same time, both claiming ownership of the same managed servers. This is not a warm standby; this must be a cold standby. The original administration server must be stopped or dead for the backup administration server to contact the managed servers.

Recovery of a Cluster

If you accidentally lost a member of a cluster or a whole cluster, you can use several ways to recover it.

 **+ Most preferable way to recover**

- Undo the changes in the Change Center.
- Reenter the configuration changes that you made.
- Use the configuration archive to go back one or two versions.
- Recover the configuration.
- Recover the domain.
- Recover WebLogic.
- Perform a full recovery.

- Least preferable way to recover

 ORACLE

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

Restoring OPMN-Managed Components to a New Computer

1. Use the methods described earlier to recover the files as though this was the same host.
2. Update the registration of the Oracle instance with the administration server using:
`updateinstanceregistration`
3. Update the registration of the component with the administration server using:
`updatecomponentregistration`
4. Edit the `targets.xml` file for Fusion Middleware Control.
5. Edit `emd` files for Enterprise Management Agent.
6. Restart the EM Agent.

ORACLE

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

Restoring OPMN-Managed Components to a New Computer

The syntax for the command to update the instance registration is:

```
opmnctl updateinstanceregistration -adminHost new_host
```

This command updates OPMN's `instance.properties` file with the new host name.

The syntax for the command to update component registration on the new host depends on the components that you are updating. For example, to update the registration for Oracle Virtual Directory, use the following command:

```
opmnctl updatecomponentregistration -Host new_host  
-Port nonSSLPort  
-componentName ovd1  
-componentType OVD
```

For the `targets.xml` file located in

`<MW_HOME>/user_projects/domains/domain_name/servers/AdminServer/sysman/state`, change the host name to the new host name.

To recover the EM Agent, edit the following files to change the host name:

```
<ORACLE_INSTANCE>/EMAGENT/emAgnt_instname/sysman/emd/targets.xml  
<ORACLE_INSTANCE>/EMAGENT/emAgnt_instname/sysman/config/emd.properties
```

If the component is Web Cache, you also need to edit the host name in:

```
<ORACLE_INSTANCE>/config/WebCache/webcache_name/webcache.xml
```

Quiz

What mode must the Middleware software be in to perform a full backup?

- a. Online
- b. Offline
- c. Either online or offline
- d. Neither. A full backup is technically impossible.

ORACLE

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

Quiz

What is another name for an inconsistent backup?

- a. Hot
- b. Cold
- c. Either online or offline
- d. Broken. If it is inconsistent, there is something wrong with it.

ORACLE

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

Quiz

When making a TAR backup in UNIX, what is a key point to remember?

- a. Make it from the lowest directory possible, as far from root as practical.
- b. Make sure that you perform the backup signed on as the owner of the Middleware Home directory.
- c. Make sure that you preserve the original owner, group, and permissions.
- d. Make sure that all Middleware processes are stopped.

ORACLE

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

Quiz

The configuration archive is enabled by default.

- a. True
- b. False

ORACLE

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

Quiz

What happens if you have a backup administration server?

- a. You are allowed to have only one administration server. If it fails, the managed servers run in MSI mode until your one administration server comes back.
- b. It runs simultaneously with the primary administration server in a load-sharing mode.
- c. It can run in a warm standby keeping itself in sync with the main administration server.
- d. It must be in cold standby and you have to sync it with the main administration server manually.



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

Answer: d

You can have only one administration server at a time; the backup administration server must be cold.

Summary

In this lesson, you should have learned how to:

- Recommend a backup and recovery strategy weighing convenience against risk
- Perform a full offline backup and recovery of all components using OS copy tools
- Perform an online domain backup and recovery of the configuration
- Perform an Instance home backup and recovery for Oracle HTTP Server and Web Cache



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

Practice 21 Overview: Backing Up and Restoring Configuration and Data

This practice covers the following topics:

- Backing up an Oracle WebLogic domain
- Backing up an Oracle HTTP Server installation
- Restoring an Oracle WebLogic domain
- Restoring an Oracle HTTP Server installation



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

Practice 21 Overview: Backing Up and Restoring Configuration and Data

See Appendix A for the complete steps to do the practice.

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

Carl McGruder (camcgruder@deloitte.com) has a non-transferable
license to use this Student Guide.

Index

Carl McGruder (camcgruder@deloitte.com) has a non-transferable
license to use this Student Guide.

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

Carl McGruder (camcgruder@deloitte.com) has a non-transferable
license to use this Student Guide.

A

Apache 2-9, 4-16, 4-26, 6-5, 6-20, 10-2, 10-10, 10-11, 16-11, 16-17, 16-21, 18-15, 21-5
API 3-6, 3-8, 3-14, 3-15, 3-18, 3-19, 3-22, 7-14, 10-8, 10-11, 12-18, 13-30, 14-4, 14-7, 14-8, 15-11, 16-5, 17-9, 19-6
autodeploy 5-35, 11-2, 11-11, 11-18, 11-30, 11-31, 11-48

B

backup 1-2, 1-7, 2-4, 7-16, 18-10, 18-13, 18-19, 18-40, 19-22, 19-23, 21-1 -- 21-3, 21-5 -- 21-16, 21-18, 21-19, 21-21, 21-24, 21-27 -- 21-29, 21-31, 21-32
BEA 4-16, 4-21, 4-23, 6-9, 6-19, 10-16

C

CA 19-9, 19-10, 20-5, 20-8, 20-10, 20-12, 20-13, 20-14, 20-18
cache 2-4, 3-25, 5-28, 5-47, 9-16, 9-18, 12-21, 12-22, 12-23, 12-31, 12-32, 18-25, 18-44, 19-23, 20-12, 20-32, 21-5
CCI 3-22
CLI 5-46
cluster 1-4, 3-26, 5-6, 5-8, 5-9, 5-17, 5-18, 5-19, 5-20, 7-11, 7-16, 7-35, 8-8, 8-14, 8-15, 8-18, 8-32, 9-7, 9-8, 9-41, 10-17, 10-25, 11-43, 13-4, 14-22, 15-13, 15-17, 15-19, 15-20, 15-21, 15-23, 15-24, 15-29, 15-30, 15-38, 16-2, 16-3, 16-4, 16-5, 16-6, 16-7, 16-9, 16-10, 16-12, 16-13, 16-15 -- 16-20, 16-22 -- 16-35, 17-2 -- 17-23, 17-26, 17-27, 18-2 -- 18-7, 18-9 -- 18-16, 18-18 -- 18-21, 18-23, 18-24, 18-29, 18-33, 18-35, 18-37 -- 18-44, 18-49, 18-50, 20-32, 21-4, 21-7, 21-11, 21-25

C

Cluster 1-4, 1-5, 5-5 -- 5-8, 5-11, 5-15, 5-17 -- 5-19, 5-32, 6-15, 7-33, 8-8, 8-14, 8-18, 10-17, 10-23, 10-24, 11-4, 11-10, 11-13, 13-15, 14-12, 16-3, 16-4, 16-7 -- 16-11, 16-12, 16-13, 16-14 -- 16-18, 16-20, 16-24, 16-25, 16-26, 16-27, 17-1, 17-3, 17-7, 17-9 -- 17-17, 17-19, 17-20, 17-25, 18-4 -- 18-8, 18-10, 18-11, 18-13, 18-18, 18-20, 18-22 -- 18-24, 18-32, 18-38, 18-40, 21-25, 7-5, 9-5, 9-7, 9-8, 16-28, 16-29, 16-34, 17-5

Coherence 2-11, 4-5, 4-9, 4-15, 4-25, 15-36

Commons 10-5, 10-8, 10-11

CORBA 3-8, 3-18

D

deploy 1-2, 2-8, 4-27, 5-8, 5-35, 7-5, 11-2, 11-4, 11-6, 11-7, 11-9, 11-10, 11-13, 11-15, 11-16, 11-22 -- 11-27, 11-31, 11-33, 11-46 -- 11-48, 12-2, 12-3, 12-5, 12-7, 12-28, 12-29, 12-34 -- 12-36, 12-40, 13-4, 13-5, 13-15, 13-16, 13-18, 13-20, 13-28, 13-31, 13-39, 14-5, 14-6, 14-12, 14-19, 14-22, 15-18, 15-21, 15-38, 16-12, 16-32, 17-2, 18-2, 18-4, 18-5, 18-6, 19-14, 19-27

Derby 4-9, 4-16, 4-25, 4-26, 5-40, 5-51

DMZ 11-35, 16-12, 16-16, 16-18

DNS 3-15, 5-9, 5-18, 8-4, 8-10, 8-19, 9-10, 9-13, 12-13, 14-18, 16-16, 17-4, 17-6, 17-7, 17-12, 18-38

DTDs 12-32

E

EAR 5-35, 12-7, 12-27, 12-30, 12-32 -- 12-35, 13-26, 13-27, 13-36, 15-18, 15-19, 15-21, 19-29

Eclipse 3-10, 11-5, 13-8, 13-22

EIS 3-22

E

EJB 1-3, 3-8, 3-13, 3-15, 3-23, 3-30, 4-5, 5-18, 5-35, 12-3,
12-5, 12-10, 12-17 -- 12-31, 12-33, 12-34, 12-35, 12-38, 13-4,
13-27, 13-30, 16-3, 16-4, 16-6, 16-8, 16-12 -- 16-18, 16-30, 16-34,
17-7, 18-2, 18-3, 18-8, 18-19, 18-32 -- 18-35, 18-37 -- 18-41, 18-43,
18-44, 19-14, 19-15, 19-26, 19-29, 19-30, 20-7
Extend 2-8, 2-10, 5-10, 5-12, 5-22, 6-4, 6-5, 6-7 , 6-11 -- 6-13, 6-22, 6-17, 7-23, 7-26,
21-22

F

FMW 2-9, 2-13, 4-7, 5-11, 8-24

G

GUI 1-5, 3-6, 4-2, 4-3, 4-4, 4-6, 4-13, 4-14, 4-24, 5-5,
5-46, 6-6, 8-7, 14-37

H

heartbeat 16-25, 16-27, 17-19, 21-23
HTML 2-8, 3-9, 3-10, 3-11, 3-12, 3-24, 4-23, 4-27, 7-9, 11-35,
11-38, 11-40, 11-41, 12-4, 12-5, 12-8, 16-12, 19-34, 19-35
HTTP 1-4, 2-9, 2-13, 3-8, 3-9, 3-23, 3-24, 3-26, 3-27, 3-30, 4-22, 5-11, 5-16, 5-19, 5-20,
5-46, 6-20, 7-6, 7-8, 8-14, 9-7, 10-4, 10-6, 10-10, 11-34 -- 11-40, 11-42, 12-4,
12-11, 12-13, 12-14, 12-31, 12-41, 13-25, 13-27, 13-30, 16-3, 16-4, 16-6 -- 16-8, 16-11,
16-12, 16-14, 16-15, 16-17 -- 16-23, 16-30, 17-2, 17-4, 17-16,
17-21 -- 17-23, 17-27, 18-2, 18-3, 18-8, 18-9, 18-10, 18-11, 18-12, 18-14 -- 18-16, 18-18
-- 18-21, 18-23, 18-24, 18-28, 18-31, 18-32, 18-48, 19-30, 20-6,
20-27, 20-28, 20-30, 21-4 -- 21-6, 21-12, 21-13, 21-16, 21-32, 21-33

I

idempotent 16-8, 16-11, 18-33, 18-36, 18-44
IIOP 3-23, 20-24
install 4-2, 4-4 -- 4-6, 4-9, 4-11, 4-25, 4-26, 4-28,
4-29, 6-17, 6-22, 11-2, 11-10 -- 11-12, 11-14, 13-8, 13-15, 16-19,
17-4, 19-5, 19-9, 19-29, 20-10, 20-24

I
Install 1-4, 2-12, 4-2, 4-4, 4-31, 6-20, 6-22, 8-24, 11-7,
11-9, 11-12, 11-13, 13-12, 13-15, 20-10, 21-24

J

JAAS 3-8, 3-20, 19-6
Jakarta 10-11
JAR 1-2, 3-12, 4-15, 4-16, 4-17, 4-22, 4-26, 4-29, 5-34, 5-35,
5-39, 5-40, 6-4, 6-8, 6-17, 7-16, 7-26, 7-32, 8-26, 11-6, 11-32,
11-47, 12-6, 12-8, 12-27, 12-33, 12-34, 19-29, 21-15
JCA 3-22, 5-39, 13-26, 13-30
JCP 3-6, 3-8
JDBC 1-7, 2-8, 3-8, 3-14, 3-15, 3-17, 3-21, 3-22, 4-16, 4-17,
5-11, 5-12, 5-15, 5-21, 5-22, 5-23, 5-24, 5-25, 5-26, 6-5, 6-13,
6-15, 6-18, 8-21, 10-4, 10-6, 11-5, 11-22, 12-26, 12-28, 12-29, 12-32,
13-5, 13-26, 13-30, 14-1 -- 14-14, 14-16 -- 14-20,
14-22, 14-24, 14-26, 14-27, 14-29, 14-30, 14-33, 14-36, 14-37, 15-7, 15-9,
15-15, 15-36, 15-39, 15-40, 16-4, 16-6, 16-11, 16-12, 18-2, 18-9, 18-16,
18-23 -- 18-28, 18-48, 18-49, 19-33, 21-12
JDeveloper 3-10, 6-2, 6-16, 6-17, 6-26, 11-5, 13-8, 13-22, 19-5
JMS 1-3, 1-7, 3-8, 3-15, 3-19, 4-5, 5-11, 5-12, 5-15, 5-22,
5-27, 5-28, 5-32, 6-5, 6-13, 6-18, 10-4, 10-6, 10-19, 11-5, 11-22,
12-26, 12-28, 12-29, 12-32, 13-26, 13-30, 13-39, 15-1 -- 15-4,
15-7 -- 15-33, 15-35 -- 15-42, 15-44, 15-45 -- 15-52,
16-4, 16-6, 16-11, 16-29, 19-29, 19-33, 20-26, 21-4, 21-5
JMX 3-8, 3-21, 5-7, 7-3, 7-14, 7-17, 7-18, 7-26, 7-35, 7-36,
7-39, 7-44, 9-14, 11-32, 17-9
JNDI 3-8, 3-15, 3-16, 3-17, 5-23, 12-21, 12-23, 13-4, 13-30, 14-2,
14-5, 14-12, 14-15, 14-16, 14-23, 14-24, 14-32, 14-35, 15-9, 15-10, 15-22,
15-23, 15-24, 15-26, 15-30, 15-42, 16-25, 16-27, 16-31, 16-32, 17-19, 19-26,
20-7
JPA 3-14, 4-25, 12-18, 12-20

J

JRMP 3-23
jsf 11-24
JSTL 3-11, 11-24
JTA 3-8, 3-18, 8-21, 15-11, 15-23, 16-5
JTS 3-18, 10-4, 13-30
JWS 12-14
Jython 2-10, 6-18, 7-22 -- 7-25, 7-30, 11-26

L

LDAP 3-15, 5-13, 5-30, 5-51, 7-16, 12-15, 14-35, 19-12, 19-22, 19-23,
19-24, 19-29, 21-11, 21-12
log4j 10-2, 10-8, 10-11

M

MBean 2-10, 3-21, 7-27, 7-35, 7-36, 7-39, 8-5, 8-6, 8-14, 9-9,
10-7, 11-33, 14-24, 17-17, 18-31
MIME 3-9, 11-36, 11-38, 12-10, 12-11
mod_wl_ohs 2-9, 11-42, 11-43, 16-20, 16-21, 16-22, 17-21, 18-15
MSI 8-2, 8-3, 8-23, 8-27, 8-28, 8-29, 8-30, 8-37, 8-38, 9-5,
9-18, 21-23, 21-31
multicast 5-17, 5-18, 10-25, 16-25 -- 16-29, 16-32,
17-4, 17-6, 17-10, 17-11, 17-13, 17-19, 17-25

N

NIC 10-26, 10-27

O

ODBC 14-4
OHS 2-9, 2-13, 3-26, 6-20, 11-42, 11-43, 11-44, 16-20, 16-21, 16-22,
17-21, 17-22, 17-23, 17-24, 21-13, 21-16
OID 2-12, 2-13
OPMN 2-10, 2-13, 5-46, 17-22, 21-5, 21-13
opmnctl 2-10, 11-44, 17-22, 17-23, 21-13, 21-16, 21-21, 21-26
OPSS 19-5

P

PAM 3-20

P

plug-in 3-26, 11-43, 16-3, 16-4, 16-7, 16-17, 16-19, 16-20, 16-21, 16-22, 18-9, 18-10, 18-12, 18-14, 18-15
PointBase 4-9, 4-11, 4-16, 4-18, 4-22, 4-26, 5-30, 13-5, 14-9, 14-26, 14-28
proxies 3-25, 5-19, 16-23
proxy 2-9, 3-25 -- 3-27, 4-16, 4-17, 5-11, 5-19, 5-20, 7-6, 11-43, 16-3, 16-4, 16-7, 16-12, 16-17, 16-18, 16-19, 16-20, 16-23, 17-4, 17-5, 17-16, 17-21, 18-9, 18-10, 18-12, 18-14 -- 18-16, 18-23, 18-28, 20-6 16-7, 16-9, 16-17 -- 16-21, 16-23, 17-21, 18-12,

R

RAC 14-9, 14-18, 21-11
RAR 3-22, 5-35, 12-25, 12-27, 13-30
RCU 6-20, 6-21, 6-22, 21-4, 21-5
recover 1-5, 9-21, 18-9, 21-2, 21-5, 21-7, 21-9, 21-13, 21-17, 21-19, 21-21, 21-22, 21-23, 21-24, 21-25, 21-26
Recovery 1-2, 1-7, 7-16, 9-38, 15-7, 16-5, 21-1 -- 21-3, 21-5, 21-7 -- 21-10, 21-13, 21-14, 21-18 -- 21-23, 21-25, 21-32
Restore 21-14, 21-19, 21-22, 21-2, 21-17, 21-19, 21-21
RMAN 21-5, 21-10, 21-11, 21-19
RMI 3-8, 3-15, 3-23, 5-7, 13-25, 13-30, 16-4, 16-6, 16-30, 16-32, 18-33, 18-34, 18-44

S

SAML 5-31
SCSI 21-7
setDomainEnv 4-20, 4-21, 5-44, 6-10, 7-29
SOA 2-3, 2-4, 2-5, 2-7, 2-12, 2-13, 6-2, 6-11, 6-16, 6-17, 6-20, 6-21, 6-22, 6-26, 19-5, 21-4, 21-5, 21-6, 21-11
SOAP 2-3, 3-23, 3-24, 12-14

S

SQL 2-8, 5-12, 5-25, 5-26, 5-30, 5-31, 6-6, 14-2, 14-4, 14-5,
14-9, 14-16, 15-39, 18-27
sqlplus 5-25, 21-13
SSL 5-9, 5-11, 5-16, 7-6, 7-27, 7-29, 7-34, 7-36, 8-4, 8-7,
8-15, 9-13, 9-21, 9-23, 9-34, 10-26, 11-23, 11-40, 18-15, 19-9, 19-10,
19-34, 20-2 -- 20-8, 20-10, 20-14, 20-15, 20-18 -- 20-21, 20-24, 20-34, 20-37,
20-38

T

tar 21-13, 21-14, 21-16, 21-19, 21-21, 21-22
template 2-13, 4-2, 4-16, 5-10, 5-11, 5-12, 5-21, 5-22, 5-25, 5-26,
5-33, 5-43, 5-45, 5-53, 6-2 -- 6-8, 6-13 -- 6-17, 6-24 -- 6-27, 7-26, 7-28,
7-32, 7-33, 8-24 -- 8-26, 12-6, 13-6, 13-8, 13-11, 15-26, 15-29, 21-22
TopLink 4-25

U

undeploy 11-7, 11-15, 11-21 -- 11-24, 13-2, 15-18, 16-12
unicast 5-17, 16-25, 16-26, 16-29, 16-32, 17-10, 17-19, 17-25, 18-45
update 5-7, 6-2, 6-5, 7-12, 7-17, 7-18, 7-26, 7-36, 7-39, 11-5,
11-6, 11-14 -- 11-16, 11-21, 11-22, 12-26, 15-6, 15-38, 16-27, 17-18,
18-7, 18-11, 18-38, 21-26
URL 5-16, 5-23, 5-25, 6-13, 8-4, 8-11, 11-2, 11-41, 12-11, 12-12,
12-13, 12-16, 13-34, 14-5, 14-13, 14-20, 14-25, 14-26, 16-20, 17-7, 18-12,
19-15, 19-32, 20-6

W

WAR 3-12, 5-35, 12-4, 12-5, 12-7, 12-8, 12-27, 12-34, 12-35, 13-27,
13-30
WLDF 7-8, 8-20, 8-21, 10-25, 11-22

W

WLS 1-7, 2-5, 2-11, 3-7, 3-15, 3-18, 3-23, 3-29, 4-7, 4-9, 4-10, 4-23, 4-29, 5-2, 5-45, 5-52, 6-20, 7-12, 7-14, 7-23, 7-26, 7-31, 7-33, 8-21, 8-24, 8-25, 8-32, 8-33, 9-4, 9-10, 9-13, 9-21, 10-1, 10-2, 10-11, 10-14, 10-16, 10-26, 10-29, 10-30, 11-4, 11-8, 11-31, 11-33, 11-36, 11-43, 12-6, 12-18, 12-20, 12-28, 13-19, 13-27, 13-33, 13-36, 14-9, 14-32, 15-9, 15-10, 15-13, 15-15, 15-22, 16-20, 16-22, 16-23, 16-31, 18-13, 18-15, 18-20, 18-31, 18-43, 19-2, 19-3, 19-5, 19-6, 19-7, 19-8, 19-11, 19-13, 19-16, 19-17, 19-19, 19-22, 19-26, 19-31, 19-36, 19-38, 19-40, 19-41, 20-2, 20-6, 20-7, 20-15, 20-16, 20-17, 20-19, 20-23, 20-24, 20-37
WLST 1-6, 2-10, 4-16, 5-2, 5-10, 5-39, 5-41, 5-42, 5-52, 6-4, 6-5, 6-6, 6-7, 6-18, 6-25, 7-1, 7-3, 7-14, 7-18, 7-22, 7-23, 7-25 -- 7-32, 7-34, 7-36, 7-37, 7-39, 7-43 -- 7-45, 8-2, 8-4 -- 8-7, 8-14, 8-15, 8-35, 9-2, 9-5, 9-9, 9-14, 9-15, 9-39, 9-40, 9-44, 10-5, 10-13, 11-2, 11-8, 11-9, 11-25, 11-26, 11-27, 11-33, 11-48, 13-6, 13-39, 14-6, 14-16, 14-24, 14-37, 15-18, 17-8, 17-9, 17-18, 19-36, 21-25

X

XA 5-24, 14-9, 14-16, 14-17, 15-22, 15-23, 15-40
XML 1-3, 2-3, 2-8, 3-6, 3-24, 4-4, 4-15 -- 4-17, 5-8, 6-9, 6-10, 6-19, 7-16 -- 7-18, 7-20, 7-21, 7-32, 7-35, 10-5, 10-11, 10-13, 11-5, 12-2, 12-4, 12-6, 12-7, 12-10, 12-12, 12-18, 12-21, 12-27, 12-29, 12-32, 13-4, 13-14, 13-37, 14-6, 14-14, 15-9, 15-19, 15-21, 21-12

Z

zip 11-5, 21-21

Carl McGruder (camcgruder@deloitte.com) has a non-transferable
license to use this Student Guide.

Glossary

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

Carl McGruder (camcgruder@deloitte.com) has a non-transferable
license to use this Student Guide.

Glossary/Acronyms

ACID	Atomicity, consistency, isolation, durability (for DB transactions)
ACK	acknowledgement-based
AsyncRep	Asynchronous HTTP Session Replication
API	Application programming interface
BPEL	Business Process Execution Language
CA	Certificate Authority (issues SSL certificates)
CGI	Common Gateway Interface
CLI	Command-line interface (as opposed to GUI) Call-level interface (part of JDBC)
CLV	Certificate Lookup and Validation
CSR	Certificate Signing Request
CSS	Common Security Services
CMO	Current Management Object (part of WLST)
COM	Component Object Model (see also DCOM and jCOM)
CORBA	Common Object Request Broker Architecture
DBA	Database administrator
DCOM	Distributed COM
DD	deployment descriptor
DDL	Data Definition Language (part of SQL)
DMZ	Demilitarized Zone (network between firewalls)
DNS	domain name server
DoS	denial of service
DSA	Digital Signature Algorithm
DTD	document type definition
EAR	Enterprise Archive
EE	Enterprise Edition (as opposed to Standard Edition [SE])
EJB	Enterprise JavaBeans
FIFO	First in, first out (queuing)
FMW	Fusion Middleware
GMD	guaranteed message delivery
GUI	Graphical user interface (as opposed to CLI)
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IOP	Internet Inter-ORB Protocol
IIS	Internet Information Server

J2SDK	Java 2 SDK Standard Edition
JAAS	Java Authentication and Authorization Service
JACC	Java Authorization Contract for Containers
JAR	Java Archive
Java EE	Java Platform Enterprise Edition
Java SE 6	Java Platform Standard Edition 6
JAX-WS	Java API for XML-Based Web Services
JAZN	Java AuthoriZatioN
JCA	Java EE Connector Architecture
JCE	Java Cryptography Extensions
jCOM	WLS Java-to-COM bridge
JDBC	Java Database Connectivity
JDK	Java Development Kit
JKS	Java KeyStore
JMS	Java Message Service
JMX	Java Management Extensions
JNDI	Java Naming and Directory Interface
JPA	Java Persistence API
JPS	Java Platform Security
JRF	Java Required Files (also known as Portability Layer [PL]; Oracle Web services stack for SOA and WebCenter)
JRMP	Java Remote Method Protocol
JSF	JavaServer Faces
JSP	JavaServer Pages
JSR	Java Specification Request
JSSE	Java Secure Socket Extensions
JSTL	JavaServer Pages Standard Tag Libraries
JTA	Java Transaction API
JVM	Java Virtual Machine
JWS	Java Web Service
LDAP	Lightweight Directory Access Protocol
LDIF	LDAP Data Interchange Format (readable text)
LVC	Live Virtual Class
MIME	Multipurpose Internet Mail Extensions
MSI	Managed Server Independence
NES	Netscape Enterprise Server
NIC	Network Interface Card (usually an Ethernet adapter)

O/R	object-relational
OAAM	Oracle Adaptive Access Manager
OASIS	Organization for the Advancement of Structured Information Standards
OBE	Oracle By Example. Self-paced Web-based training on OTN.
ODBC	Open Database Connectivity
OEPE	Oracle Enterprise Pack for Eclipse
OHS	Oracle HTTP Server
OID	Oracle Internet Directory
OOTB	Out Of The Box (by default; without modification)
OPMN	Oracle Process Manager and Notification Server
OPSS	Oracle Platform Security Services
ORB	Object Request Broker
OS	Operating system (examples: Windows, Linux)
OTN	Oracle Technology Network. http://otn.oracle.com
PAM	Pluggable Authentication Module
PKI	Public Key Infrastructure
PTP	Point To Point
QoS	Quality of Service
RAC	Real Application Clusters (for multihost databases)
RAR	Resource Adapter Archive
RBAC	Role-Based Access Control (part of JPS)
RCU	Repository Creation Utility
RMAN	Recovery Manager (for the database)
RMI	Remote Method Invocation
RSH	Remote Shell (as opposed to SSH)
SAF	store-and-forward
SAML	Security Assertion Markup Language
SDK	Software development kit (programming tools)
SHA	Secure Hash Algorithm
SLA	service-level agreements
SNMP	Simple Network Management Protocol
SOA	Service-Oriented Architecture
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SSH	Secure Shell (as opposed to RSH)
SSL	Secure Sockets Layer
SSPI	Security Services Provider Interface

T3	An optimized, proprietary communications protocol used to transport data between WebLogic Server and other Java programs, including clients and other WebLogic Servers.
T3S	T3 protocol using SSL
TAR	Tape Archive
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TTL	time-to-live
URL	Uniform Resource Locator
VM	Virtual Machine
WAN	Wide Area Network
WAR	Web Archive
WLDF	WebLogic Diagnostics Framework
WLS	WebLogic Server
WLST	WebLogic Scripting Tool
WSIT	Web Services Interoperability Technologies
XA	X/Open Distributed Transaction Processing (part of two-phase commit)
XACML	eXtensible Access Control Markup Language
XML	Extensible Markup Language