Oracle® Fusion Middleware

High Availability Guide

12*c* (12.1.2)

E38249-02

September 2013

Reference documentation for administrators, developers, and others that describes high availability concepts as well as administration and configuration procedures to deploy and manage Oracle Fusion Middleware with high availability requirements.

ORACLE

Oracle Fusion Middleware High Availability Guide, 12*c* (12.1.2)

E38249-02

Primary Author: Christine Ford

Contributing Author: Michael Blevins, Suvendu Ray, Peter LaQuerre

Contributor: Gururaj BS

Contents

Preface vii

Audience vii
Purpose of this Guide vii
Documentation Accessibility vii
Related Documents vii
Conventions viii

Part I Introduction to High Availability

1 Introduction and Roadmap

1.1 How to Use This Guide 1-1
1.2 New and Changed Features in This Release 1-2
1.3 What is High Availability? 1-3
1.4 High Availability Solutions 1-3
1.5 Understanding the Oracle Fusion Middleware Standard HA Topology 1-4

2 High Availability Concepts

2.1 Server Load Balancing in a High Availability Environment 2-1
2.1.1 Third-Party Load Balancer Requirements 2-2
2.1.2 Third-Party Load Balancer Configuration 2-3
2.1.3 Server Load Balancing with Oracle HTTP Server or Oracle Traffic Director 2-3
2.2 Application Failover 2-4
2.3 Real Application Clusters 2-5
2.4 Coherence Clusters and High Availability 2-5
2.5 Disaster Recovery 2-6
2.6 Install Time Configuration 2-6
2.6.1 Domain (Topology) Profiles 2-6
2.6.2 Persistence Profiles 2-7
2.7 Application and Service Failover 2-7
2.7.1 Whole Server Migration 2-8
2.7.2 Automatic Service Migration 2-8
2.8 Roadmap for Setting Up a High Availability Topology 2-9

Part II Creating a High Availability Environment

3 Using Shared Storage

3.1 Overview of Shared Storage ... 3-1
3.2 Shared Storage Prerequisites ... 3-2
3.3 Using Shared Storage for Binary (Oracle Home) Directories ... 3-2
3.3.1 About the Binary (Oracle Home) Directories ... 3-3
3.3.2 About Using Redundant Binary (Oracle Home) Directories ... 3-3
3.4 Using Shared Storage for Domain Configuration Files ... 3-3
3.4.1 About Oracle WebLogic Server Administration and Managed Server Domain Configuration Files ... 3-4
3.4.2 Shared Storage Considerations for Administration and Managed Server Domain Configuration Files ... 3-4
3.5 Shared Storage Requirements for JMS Stores and JTA Logs ... 3-4
3.6 Directory Structure and Configurations ... 3-4

4 Database Considerations

4.1 About Oracle Real Application Clusters ... 4-1
4.2 About RAC Database Connections and Failover ... 4-2
4.2.1 About XA Transactions ... 4-2
4.3 About Data Sources ... 4-2
4.3.1 Active GridLink Data Sources ... 4-3
4.3.2 Multi Data Sources ... 4-4
4.4 Configuring Active GridLink Data Sources with Oracle RAC ... 4-4
4.4.1 Requirements ... 4-4
4.4.2 Configuring Component Data Sources as Active GridLink Data Sources ... 4-5
4.4.3 Using Single Client Access Name (SCAN) Addresses for Hosts and Ports ... 4-6
4.5 Configuring Multi Data Sources ... 4-6
4.5.1 Configuring Multi Data Sources with Oracle RAC ... 4-6
4.5.1.1 Requirements ... 4-7
4.5.1.2 Configuring Component Data Sources as Multi Data Sources ... 4-7
4.5.1.3 Modifying or Creating Multi Data Sources After Initial Configuration ... 4-7
4.5.2 Configuring Multi Data Sources for MDS Repositories ... 4-9

5 JMS and JTA High Availability

5.1 About JMS and JTA Services for High Availability ... 5-1
5.2 Configuring JMS and JTA Services for High Availability ... 5-2
5.3 User-Preferred Servers and Candidate Servers ... 5-2
5.4 Considerations for Using File Persistence (WebLogic JMS) ... 5-2
5.4.1 Considerations for Using File Stores on NFS ... 5-3
5.5 Configuring Schemas for Transactional Recovery Privileges ... 5-6

6 Scaling Out a Topology (Machine Scale Out)

6.1 About Machine Scale Out ... 6-1
6.2 Roadmap for Scaling Out Your Topology ... 6-2
6.3 Optional Scale Out Procedure ... 6-2
6.4 About Scale Out Prerequisites ... 6-3
6.5 Resource Requirements ... 6-3

6.6 Creating a New Machine 6-4
6.6.1 Shutting Down the Managed Server 6-4
6.6.2 Creating a New Machine 6-4
6.6.3 Assigning Managed Servers to the New Machine 6-5
6.7 Packing the Domain on APPHOST1 6-5
6.8 Preparing the New Machine 6-6
6.9 Running Unpack to Transfer the Template 6-6
6.10 Starting the Node Manager 6-7
6.11 Starting the Managed Servers 6-7
6.12 Verifying Machine Scale Out 6-7

Part III Component Procedures

7 Configuring High Availability for Web Tier Components

7.1 Oracle HTTP Server and High Availability Concepts 7-1
7.2 Oracle HTTP Server Single-Instance Characteristics 7-2
7.2.1 Oracle HTTP Server and Oracle WebLogic Server 7-2
7.3 Oracle HTTP Server Startup and Shutdown Lifecycle 7-3
7.4 Starting and Stopping Oracle HTTP Server 7-3
7.5 Oracle HTTP Server High Availability Architecture and Failover Considerations 7-3
7.6 Oracle HTTP Server Protection from Failures and Expected Behaviors 7-4
7.7 Oracle HTTP Server Cluster-Wide Configuration Changes 7-5
7.8 Configuring Oracle HTTP Server for High Availability 7-5
7.8.1 Prerequisites 7-5
7.8.1.1 Configuring the Load Balancer 7-6
7.8.1.2 Installing Oracle HTTP Server on WEBHOST1 7-6
7.8.1.3 Configuring Virtual Host(s) 7-7
7.8.1.4 Configuring mod_wl_ohs.conf 7-7
7.8.2 Installing Oracle HTTP Server on WEBHOST2 7-8
7.8.3 Configuring and Validating the OHS High Availability Deployment 7-8
7.8.3.1 Configuring Virtual Host(s) 7-8
7.8.3.2 Validating the Oracle HTTP Server Configuration 7-9

8 Configuring High Availability for Oracle Application Development Framework

8.1 Oracle ADF High Availability Considerations 8-1
8.1.1 Oracle ADF Scope and Session State 8-1
8.1.2 Oracle ADF Failover and Expected Behavior 8-2
8.1.3 Configuring the ADF Application Module for Oracle RAC 8-2
8.2 Configuring Oracle ADF for High Availability 8-3
8.2.1 Configuring Application Modules 8-3
8.2.2 Configuring weblogic.xml 8-4
8.2.3 Configuring adf-config.xml 8-5
8.2.4 Configuring org.apache.myfaces.trinidad.CHECK_FILE_MODIFICATION 8-6
8.3 Troubleshooting Oracle ADF High Availability 8-6
8.3.1 Troubleshooting Oracle ADF Development Issues 8-6
8.3.2 Troubleshooting Oracle ADF Deployment Issues 8-7

8.3.2.1 Verifying the JRF Runtime Installation 8-7
8.3.2.2 Verifying the Success of All Application Deployments 8-7
8.3.3 Troubleshooting Oracle ADF Replication and Failover Issues 8-7

9 Configuring High Availability for Other Components

9.1 Deploying the Oracle Virtual Assembly Builder Deployer 9-1
9.2 Deploying Oracle Data Integrator 9-1
9.2.1 Oracle RAC Retry Connectivity for Source and Target Connections 9-1
9.2.2 Configuring Repository Connections to Oracle RAC 9-1

Preface

This preface contains these sections:

- Audience
- Documentation Accessibility
- Related Documents
- Conventions

Audience

The *High Availability Guide* is intended for administrators, developers, and others whose role is to deploy and manage Oracle Fusion Middleware with high availability requirements.

Purpose of this Guide

The purpose of this guide is to serve as a reference document to set up a highly available environment. Use this guide in conjunction with your product's installation and administration guides.

Documentation Accessibility

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

Access to Oracle Support

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

Related Documents

For more information, see these Oracle resources:

- *Understanding Oracle Fusion Middleware Concepts*
- *Administering Oracle Fusion Middleware*

- *Oracle Fusion Middleware Tuning Performance Guide*

Conventions

The following text conventions are used in this document:

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

Part I

Introduction to High Availability

Part I contains the following chapters:

- Chapter 1, "Introduction and Roadmap"
- Chapter 2, "High Availability Concepts"

1 Introduction and Roadmap

This chapter includes introductory information on how and why to use this guide and high availability environments.

This chapter includes the following topics:

- Section 1.1, "How to Use This Guide."
- Section 1.2, "New and Changed Features in This Release"
- Section 1.3, "What is High Availability?"
- Section 1.4, "High Availability Solutions"
- Section 1.5, "Understanding the Oracle Fusion Middleware Standard HA Topology"

1.1 How to Use This Guide

Use this document as a reference guide for information on high availability concepts and tasks as you set up a highly available environment.

Before you use this guide, you must have a standard installation topology set up for your product. This is the required starting point for setting up high availability. See the topics "Understanding the Oracle Fusion Middleware Infrastructure Standard Installation Topology" and "Roadmap for Installing and Configuring the Standard Installation Topology" to set up the standard installation topology.

Table 1–1 describes tasks to set up a highly available environment and resources for information that is not in this guide.

Table 1–1 Setting up a Highly Available Environment

Task	Description	For more information
Performing administrative tasks and preparing your environment	Common tasks to perform on a newly-created domain.	See the topic "Administering and Preparing your WebLogic Domain for High Availability" in your product installation guide.
Planning your WebLogic Server Installation	Covers understanding your topology and determining the distribution, components and features you need	See the guide *Planning an Installation of Oracle Fusion Middleware*

Table 1–1 (Cont.) Setting up a Highly Available Environment

Task	Description	For more information
Installing the WebLogic Server Software	Describes how to start the installation process and progress through the installation screens	See the topic "Installing the Oracle Fusion Middleware Infrastructure Software" in your product installation guide.
Configuring a domain	Creating and configuring a domain	See the topic "Configuring your Oracle Fusion Middleware Infrastructure Domain" in your product installation guide.
Managing Oracle Fusion Middleware	Includes how to: start and stop, change ports, deploy applications, and back up and recover Oracle Fusion Middleware.	See *Oracle Fusion Middleware Administrator's Guide*
Monitoring and optimizing performance in the Oracle Fusion Middleware environment.	For components that impact performance, use multiple components for optimal performance, and design applications for performance.	See *Oracle Fusion Middleware Tuning Performance Guide*
Setting up a product-specific enterprise deployment	Oracle best practices blueprints based on proven Oracle high availability and security technologies and recommendations for a product-specific enterprise deployment.	See your product's Enterprise Deployment Guide
Administering the product environment	To deploy, manage, monitor, and configure applications using the product.	See your product's Administrator's Guide
Configuring Node Manager	Node Manager enables you to start, shut down, and restart the Administration Server and Managed Server instances from a remote location, making this an essential utility for any high availability environment.	See the guide *Administering Node Manager for Oracle WebLogic Server*

1.2 New and Changed Features in This Release

Oracle Fusion Middleware 12c Release 1 (12.1.2) includes the following new and changed concepts and features from previous Oracle Fusion Middleware releases:

- Middleware topology changes. See "Understanding Middleware Architecture Design."
- Support for a domain-based Node Manager. See "What is Node Manager?"
- Managing system components. You no longer use OPMN agent and `opmnctl` command line utility in Oracle Fusion Middleware. You manage system

components with the WebLogic Management Framework, which includes management tools such as WLST. For more information, see "What is the WebLogic Management Framework?" in the guide *Oracle Fusion Middleware Understanding Oracle Fusion Middleware* and WLST Command and Variable Reference.

- Redefining of the Oracle home and elimination of the Middleware home. See "New and Deprecated Terminology for 12c Release 1 (12.1.2)" in *Understanding Oracle Fusion Middleware Concepts*.
- Coherence replaces the Java Object Cache. See Section 2.4, "Coherence Clusters and High Availability".

See Also: For a comprehensive list of new and deprecated:

- **WebLogic Server features** in this release, see *Oracle Fusion Middleware What's New in Oracle WebLogic Server.*
- **Terms** in this release, see "New and Deprecated Terminology for 12c Release 1 (12.1.2)" in *Understanding Oracle Fusion Middleware Concepts*

1.3 What is High Availability?

High availability is the ability of a system or device to be available when it is needed.

A high availability architecture ensures that users can access a system without loss of service. Deploying a high availability system minimizes the time when the system is down, or unavailable, and maximizes the time when it is running, or available.

High availability comes from redundant systems and components. You can categorize high availability solutions by their level of redundancy into **active-active** solutions and **active-passive** solutions.

An **active-active solution** deploys two or more active servers and can be used to improve scalability and provide high availability. In active-active deployments, all instances handle requests concurrently. Oracle recommends active-active solutions for all single-site middleware deployments. **Active-passive solutions** deploy an active instance that handles requests and a passive instance that is on standby.

1.4 High Availability Solutions

You can categorize high availability solutions into local **high availability solutions** that provide high availability in a single data center deployment, and **disaster recovery solutions**.

Local high availability solutions can protect against process, node, and media failures, as well as human errors, ensuring availability in a single data center deployment.

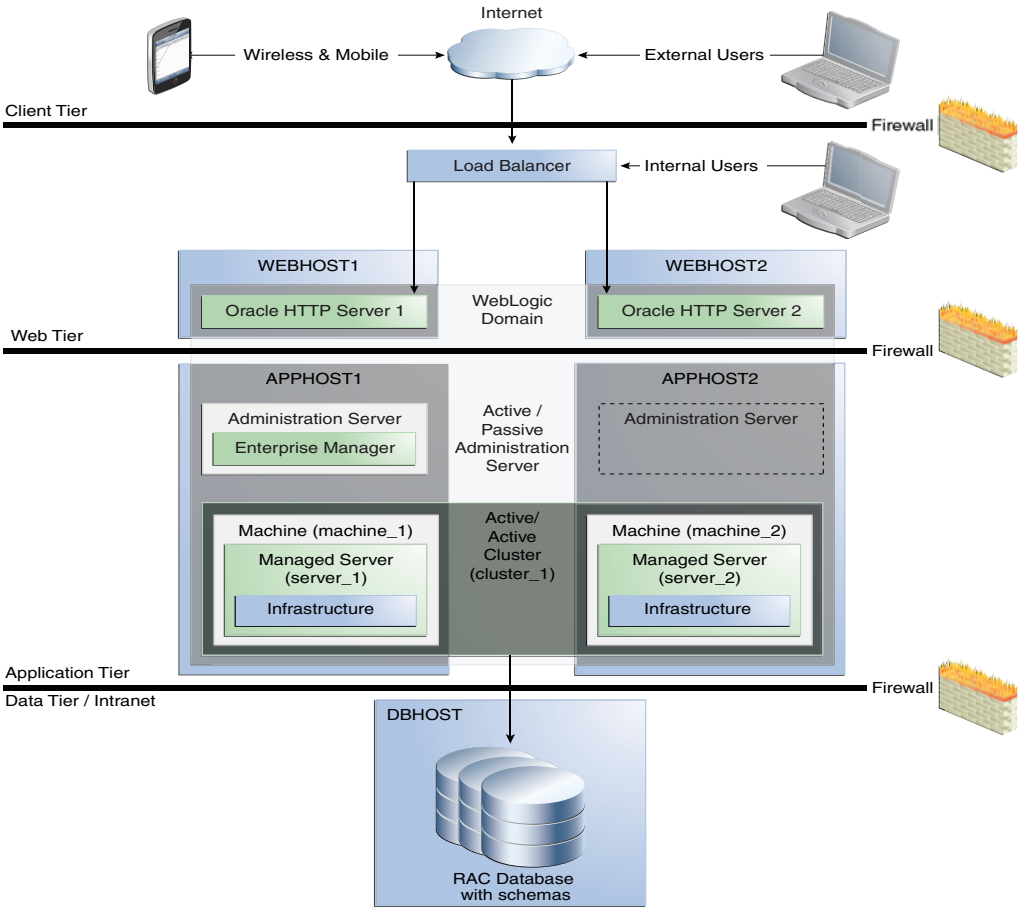
Disaster recovery solutions are usually geographically distributed deployments that protect your applications from disasters such as floods or regional network outages. You can protect against physical disasters that affect an entire data center by deploying geographically-distributed disaster recovery solutions. For detailed information about disaster recovery for Oracle Fusion Middleware components, refer to the *Oracle Fusion Middleware Disaster Recovery Guide*

1.5 Understanding the Oracle Fusion Middleware Standard HA Topology

Figure 1–1 shows the recommended standard high availability topology for a local, highly available Oracle Fusion Middleware deployment.

This deployment is consistent with the infrastructure standard installation topology and Oracle HTTP Server standard installation topology if you followed instructions in the *Oracle Fusion Middleware Installing and Configuring the Oracle Fusion Middleware Infrastructure* and *Installing and Configuring Oracle HTTP Server* guides.

Figure 1–1 Oracle Fusion Middleware Highly Available Deployment Topology (Typical Enterprise)



This topology represents a multi-tiered architecture. Users access the system from the client tier. Requests go through a hardware load balancer, which routes them to Web servers running Oracle HTTP Servers in the web tier. Web servers use Proxy Plug-in (mod_wl_ohs) to route requests to the WebLogic cluster in the application tier. Applications running on the WebLogic cluster in the application tier then interact with the database cluster in the data tier to service the request.

Table 1–2 describes elements in Figure 1–1.

Table 1–2 Description of the Elements in the Oracle Fusion Middleware Infrastructure Standard High Availability Topology

Element	Description and Links to Additional Documentation
APPHOST	Refers to the machine that hosts the application tier.

Table 1–2 (Cont.) Description of the Elements in the Oracle Fusion Middleware Infrastructure Standard High Availability Topology

Element	Description and Links to Additional Documentation
WEBHOST	Refers to the machine that hosts the web tier.
WebLogic Domain	A logically related group of Java components, in this case, the Administration Server, Managed Servers, and other related software components. For more information, see "What is an Oracle WebLogic Server Domain?" in *Understanding Oracle Fusion Middleware.*
Administration Server	The central control entity of a domain which maintains the domain's configuration objects and distributes configuration changes to Managed Servers.
Enterprise Manager	Oracle Enterprise Manager Fusion Middleware Control. This is the main tool that you use to manage a domain.
Cluster	A collection of multiple WebLogic Server instances running simultaneously and working together.
Machine	Logical representation of the computer that hosts one or more WebLogic Server instances (servers). Machines are also the logical glue between WebLogic Managed Servers and the Node Manager; to start or stop a Managed Server with Node Manager, the Managed Server must be associated with a machine.
Managed Server	Host for your applications, application components, Web services, and their associated resources. For more information, see "Oracle Enterprise Manager Fusion Middleware Control" in *Understanding Oracle Fusion Middleware*.
Infrastructure	Collection of services that includes: ■ Metadata repository (MDS) This contains metadata for Oracle Fusion Middleware components, such as the Oracle Application Developer Framework. For more information, see "What is the Metadata Repository?" in *Understanding Oracle Fusion Middleware*. ■ Oracle Application Developer Framework (Oracle ADF) ■ Oracle Web Services Manager (OWSM)

See Also:

- To view a figure of the Infrastructure Standard Installation Topology and follow a roadmap to install it, see "Understanding the Oracle Fusion Middleware Infrastructure Standard Installation Topology" in the guide *Oracle Fusion Middleware Installing and Configuring the Oracle Fusion Middleware Infrastructure*.
- To view a figure of the Oracle HTTP Server Standard Installation Topology and follow a roadmap to install it, see Introducing the Oracle HTTP Server Standard Installation Topologies in the guide *Installing and Configuring Oracle HTTP Server*.

2

High Availability Concepts

This chapter describes high availability concepts and includes the following topics:

- Section 2.1, "Server Load Balancing in a High Availability Environment"
- Section 2.2, "Application Failover"
- Section 2.3, "Real Application Clusters"
- Section 2.4, "Coherence Clusters and High Availability"
- Section 2.5, "Disaster Recovery"
- Section 2.6, "Install Time Configuration"
- Section 2.7, "Application and Service Failover"
- Section 2.8, "Roadmap for Setting Up a High Availability Topology"

For information on Oracle Fusion Middleware concepts, see the following topics in the guide *Oracle Fusion Middleware Understanding Oracle Fusion Middleware Concepts*.

Table 2–1 Oracle Fusion Middleware Concepts

For information on...	See this topic...
Oracle Home, Oracle Common, WebLogic Server Domain	"What are the Key Oracle Fusion Middleware Directories?"
WebLogic Server Domain	"What is an Oracle WebLogic Server Domain?"
Administration Server	"What is the Administration Server?"
Managed Servers and Managed Server Clusters	"Understanding Managed Servers and Managed Server Clusters"
Node Manager	"What is Node Manager?"

See Also:

"Communications in a Cluster" in *Oracle Fusion Middleware Administering Clusters for Oracle WebLogic Server*

2.1 Server Load Balancing in a High Availability Environment

Load balancing is the even distribution of jobs and associated communications across the computing and networking resources in your environment.

Typically, Oracle Fusion Middleware high availability deployments are front ended by a load balancer that you configure to distribute incoming requests using various algorithms. You can configure load balancing between different components or applications.

You can also configure load balancing between different components or applications by using Oracle HTTP Server or, on Exalogic only, Oracle Traffic Director. See Section 2.1.3, "Server Load Balancing with Oracle HTTP Server or Oracle Traffic Director" for more information.

This section includes the following topics:

- Section 2.1.1, "Third-Party Load Balancer Requirements"
- Section 2.1.2, "Third-Party Load Balancer Configuration"
- Section 2.1.3, "Server Load Balancing with Oracle HTTP Server or Oracle Traffic Director"

2.1.1 Third-Party Load Balancer Requirements

You can use third-party load balancers in your Oracle Fusion Middleware high availability deployments. Your external load balancer must have the following features:

- Ability to load-balance traffic to a pool of real servers through a virtual host name: clients access services using the virtual host name instead of using actual host names. The load balancer can then load balance requests to the servers in the pool.

  Note: Typically the load balancer can balance across Oracle HTTP Server instances and then the Oracle HTTP Servers can balance across application servers.

- Port translation configuration.
- Monitoring of ports (HTTP and HTTPS).
- Virtual servers and port configuration: ability to configure virtual server names and ports on your external load balancer, and the virtual server names and ports must meet the following requirements:
  - The load balancer should allow configuration of multiple virtual servers. For each virtual server, the load balancer should allow configuration of traffic management on more than one port. For example, for Oracle WebLogic Clusters, the load balancer must be configured with a virtual server and ports for HTTP and HTTPS traffic.
  - The virtual server names must be associated with IP addresses and be part of your DNS. Clients must be able to access the external load balancer through the virtual server names.
- Ability to detect node failures and immediately stop routing traffic to the failed node.
- Resource monitoring / port monitoring / process failure detection: the load balancer must be able to detect service and node failures through notification or some other means and to stop directing non-Oracle Net traffic to the failed node. If your external load balancer has the ability to automatically detect failures, Oracle recommends that you use it.

- Fault tolerant mode: Oracle recommends that you configure the load balancer to be in fault-tolerant mode.
- Virtual server returning to calling client: Oracle highly recommends that you configure the load balancer virtual server to return immediately to the calling client when the back-end services that it forwards traffic to are unavailable. This is preferred over the client disconnecting on its own after a timeout based on the TCP/IP settings on the client machine.
- SSL acceleration. Oracle recommends this feature but does not require it.
- Ability to Preserve the Client IP Addresses: the load balancer must have the capability to insert the original client IP address of a request in an X-Forwarded-For HTTP header to preserve the Client IP Address.

2.1.2 Third-Party Load Balancer Configuration

Detailed load balancer configuration steps depend on:

- The environment you are using the load balancer in.
- The type of load balancer you are using.

For these reasons, Oracle recommends that you follow the documentation for your own load balancer. For high-level load balancer configuration steps, see the enterprise deployment guide for the component you are working with.

2.1.3 Server Load Balancing with Oracle HTTP Server or Oracle Traffic Director

This section describes Oracle products you can use to provide load balancing.

Server Load Balancing with Oracle HTTP Server

Oracle HTTP Server is a web server with built-in WebLogic Server Proxy Plug-In module to act as HTTP front-end for one or more WebLogic servers. This built-in receives the incoming request from the client and load balances the request to one or more WebLogic Servers.

Oracle HTTP Server includes the `mod_wl_ohs` module, which routes requests to Oracle WebLogic Server. The `mod_wl_ohs` module provides the same load balancing functionality as `mod_weblogic`, the Oracle WebLogic Server Plug-in for Apache HTTP Server. See "Configuring the mod_wl_ohs Plug-In for Oracle HTTP Server" in *Oracle Fusion Middleware Using Web Server 1.1 Plug-Ins with Oracle WebLogic Server* for more information. See Section 7.8.1.4, "Configuring mod_wl_ohs.conf" for more information on the `mod_wl_ohs` module.

Server Load Balancing with Oracle Traffic Director

Oracle Traffic Director is a highly available Application Delivery Controller with WebLogic inter-operability enhancements to allow incoming requests to be efficiently throttled to one or more WebLogic server clusters. Oracle Traffic Director is a fast, reliable, and scalable layer-7 software load balancer. You can set up Oracle Traffic Director to serve as the reliable entry point for all HTTP, HTTPS and TCP traffic to application servers and web servers in the back end. Oracle Traffic Director distributes the requests that it receives from clients to servers in the back end based on the specified load-balancing method, routes the requests based on specified rules, caches frequently accessed data, prioritizes traffic, and controls the quality of service.You can use Oracle Traffic Director with Exalogic only.

2.2 Application Failover

Failover is relocating an overloaded or failed resource such as a server, disk drive, or network to its redundant or backup component.

Application failover is when an application component doing a particular job becomes unavailable for any reason and a copy of the failed component finishes the job.

Information about what has been done on a job is called **state**. WebLogic Server maintains state information using techniques called session replication and replica-aware stubs. When a component unexpectedly stops doing its job, replication techniques enable a copy of the component to pick up where the failed component stopped and finish the job.

Session Failover Requirements

Note: Oracle applications meet these session failover requirements unless a specific exception is made for an application.

For seamless application failover, an application must meet the following conditions:

- The application is in a cluster and at least one member of the application cluster is available to serve the request.
- For stateful applications, state replication is configured correctly.
- If you are using Oracle HTTP Server, the server is configured with the WebLogicCluster directive to balance among all available application instances.
- If you are using a hardware load balancer, the load balancer is:
  - Routing traffic to all available instances
  - Configured correctly with a health monitor to mark unavailable instances
  - Configured to support persistence of session state

Expected Behavior for Application Failover

If you configure the environment correctly, application users do not notice when an application instance in a cluster becomes unavailable. The sequence of events in an application failover is, for example, as follows:

1. A user makes a request and a hardware load balancer routes it to Instance A of the application.
2. Instance A of the application becomes unavailable because of node failure, process failure, or network failure.
3. The hardware load balancer marks Instance A as unavailable.
4. The user makes a subsequent request. The request is routed to Instance B.
5. Instance B is configured as a replication partner of Instance A and has the user's session state.
6. The application resumes using the session state on Instance B and the user continues working without interruption.

See Also: See the *Domain Template Reference* for information on domain and extension templates that support high availability.

See Failover and Replication in a Cluster in the guide *Administering Clusters for Oracle WebLogic Server* for information on failover and replication at the application level.

2.3 Real Application Clusters

Oracle Real Application Clusters (RAC) enable you to cluster an Oracle database. A **cluster** comprises multiple interconnected computers or servers that appear as if they are one server to end users and applications. Oracle RAC uses Oracle Clusterware for the infrastructure to bind multiple servers so they operate as a single system. Along with a collection of hardware (cluster), Oracle RAC unites the processing power of each component to become a single, robust computing environment. Oracle RAC simultaneously provides a highly scalable and highly available database for Oracle Fusion Middleware.

Every Oracle RAC instance in the cluster has equal access and authority. Node and instance failure may affect performance but does not result in downtime because the database service is available or can be made available on surviving server instances.

See Also: For more information on Oracle RAC see:

- *Oracle Fusion Middleware Administering Clusters for Oracle WebLogic Server*
- *Oracle Real Application Clusters Administration and Deployment Guide*
- *Oracle Clusterware Administration and Deployment Guide*

2.4 Coherence Clusters and High Availability

If you follow *Oracle Fusion Middleware Installing and Configuring Oracle WebLogic Server and Coherence* or a product installation guide, such as *Oracle Fusion Middleware Installing and Configuring Oracle Fusion Middleware for Oracle Application Development Framework*, the standard installation topology includes a standard Coherence cluster that serves as a starting point for additional configuration.

A **Coherence cluster** is a collection of Java Virtual Machine (JVM) processes running Coherence. In 12c, these processes are referred to as WebLogic Managed Coherence Servers. JVMs that join a cluster are called **cluster members** or **cluster nodes**. Cluster members can be:

- Dedicated storage members
- Client members that have storage disabled
- Proxy members that allow non-cluster members to access Coherence caches

Cluster members communicate using Tangosol Cluster Management Protocol (TCMP). Cluster members use TCMP for both multicast communication (broadcast) and unicast communication (point-to-point communication).

Coherence characteristics include the following:

- Each domain typically contains one Coherence Cluster.

- Each managed Coherence server in a domain is associated with a Coherence cluster, defined through a Coherence Cluster System Resource.
- Each application includes its Coherence configuration in a Grid Archive (GAR) file. The GAR file is deployed with the application and to all dedicated storage nodes.

All Fusion Middleware applications that use Coherence use the cluster associated with the managed Coherence server and deploy their GAR files co-located with their applications. Table 2–2 provides additional sources of information about Coherence.

Table 2–2 Coherence and Coherence Clusters

For information on...	See this topic...
Coherence concepts and features	"Introduction to Coherence" in *Oracle Fusion Middleware Developing Applications with Oracle Coherence*
Creating Coherence clusters	"Setting Up a WebLogic Server Domain Topology for Coherence" in *Coherence Administrator's Guide*
Configuring a Coherence Cluster	"Configuring and Managing Coherence Clusters" in *Administering Clusters for Oracle WebLogic Server*

2.5 Disaster Recovery

For maximum availability, you may need to deploy services at different geographical locations to protect against entire site failures due to unforeseen disasters and natural calamities. Oracle Fusion Middleware products support the configuration of a geographically separate standby site to act as a backup. Applications and services can fail over to this backup in the event of natural or unplanned outages at a production site.

For detailed information about disaster recovery for Oracle Fusion Middleware components, refer to *Oracle Fusion Middleware Disaster Recovery Guide*.

2.6 Install Time Configuration

This section includes the following topics:

- Section 2.6.1, "Domain (Topology) Profiles"
- Section 2.6.2, "Persistence Profiles"

2.6.1 Domain (Topology) Profiles

You use the Configuration Wizard or WebLogic Scripting Tool (offline) to set up domains. See the guide Oracle Fusion Middleware Creating WebLogic Domains Using the Configuration Wizard for procedures to create, update, and configure domains.

All 12*c* (12.1.2) installation guides provide instructions for setting up a single machine, multi-server domain using the expanded profile. See one of the following guides for more information:

- *Oracle Fusion Middleware Installing and Configuring Oracle HTTP Server*
- *Oracle WebLogic Server Installing and Configuring Fusion Middleware for Oracle ADF Applications*

2.6.2 Persistence Profiles

Persistence profiles are a collection of settings intended to meet a specific persistence environment. There are two persistence profiles for Expanded profiles: database and file based.

Table 2–3 shows persistence types for database and file persistence profiles.

Table 2–3 Persistence Types for Database and File Persistence Profiles

Component/Service	Database Persistence Profile	File Persistence Profile
JMS	AQ in Database Store	WebLogic Server JMS in File Store
JTA	JTA in Database Store	JTA in File Store
OPSS	Database Store	Database Store
MDS	Database Store	Database Store
Service Table	Database Store	Database Store
Failover	Automatic Service Migration	Whole Server Migration

Although you can "mix & match" a component or service with the persistence profile, the persistence type groups in Table 2–3 work together optimally. Oracle recommends that you use all options consistently within their respective profile.

> **See Also:** See "Interoperability with Supported Databases" in the *Interoperability and Compatibility Guide* for database version requirements for selected products and features.

Post-Configuration Defaults

For the expanded domain configuration, the database persistence mode is configured by default. For compact domain configuration, file persistence is the default.

> **Note:** Some products may have specific requirements for shared file stores; Oracle recommends that you refer to your product's requirements for details.

Table 2–4 describes additional sources of information. To configure file persistence after an expanded domain configuration, such as moving from database persistence to file persistence, see Chapter 5, "JMS and JTA High Availability".

Table 2–4 Domain Configuration Topics

For additional information on...	See this topic...
Shared file systems for use with the file persistence profile	Chapter 3, "Using Shared Storage"
JMS and JTA	Section 5.2, "Configuring JMS and JTA Services for High Availability"
Failover	Section 2.2, "Application Failover"

2.7 Application and Service Failover

Migration in WebLogic Server is the process of moving a clustered WebLogic Server instance or a component running on a clustered instance elsewhere if failure occurs.

Whole server migration occurs when the server instance migrates to a different physical system upon failure. With **service-level migration**, the services move to a different server instance within the cluster.

This section describes server and service failover for JMS and JTA.

Topics in this section include:

- Section 2.7.1, "Whole Server Migration"
- Section 2.7.2, "Automatic Service Migration"

2.7.1 Whole Server Migration

A WebLogic Server cluster provides high availability and failover by duplicating an object or service on redundant servers in the cluster. However, some services, such as JMS servers and the JTA transaction recovery service, are designed with the assumption that there is only one active instance of the service running in a cluster at any given time. These types of services are referred to as **pinned services** because they remain active on only one server instance at a time.

In a WebLogic Server cluster, most services deploy homogeneously on all server instances in the cluster, enabling transparent failover from one server to another. However, pinned services such as JMS and the JTA transaction recovery system are targeted at individual server instances within a cluster. For these services, WebLogic Server supports failure recovery with migration instead of failover.

WebLogic Server provides a feature for making JMS and the JTA transaction system highly available: *migratable servers*. Migratable servers provide for both automatic and manual migration at the server-level, rather than the service level.

When a migratable server becomes unavailable for any reason, for example, if it hangs, loses network connectivity, or its host system fails—migration is automatic. Upon failure, a migratable server automatically restarts on the same system if possible. If the migratable server cannot restart on the system it failed on, it migrates to another system. In addition, an administrator can manually initiate migration of a server instance.

See Also: See Whole Server Migration in *Oracle Fusion Middleware Administering Clusters for Oracle WebLogic Server* for more information on preparing for automatic whole server migration, configuring automatic whole server migration, and server migration processes and communications.

See Section 5, "JMS and JTA High Availability" for more details on JMS and JTA services.

2.7.2 Automatic Service Migration

Service-level migration in WebLogic Server is the process of moving pinned services from one server instance to a different available server instance within the cluster.

You can configure JMS and JTA services for high availability by using migratable targets. A **migratable target** is a special target that can migrate from one server in a cluster to another. As such, a migratable target provides a way to group migratable services that should move together. High availability is achieved by migrating a migratable target from one clustered server to another when a problem occurs on the original server. When the migratable target is migrated, all services hosted by that target are migrated.

See Also: For more information, see "Service Migration" in the guide *Oracle Fusion Middleware Administering Clusters for Oracle*, which includes the following topics:

- "Understanding the Service Migration Framework"
- "Pre-Migration Requirements"
- "Roadmap for Configuring Automatic Migration of JMS-related Services"
- "Roadmap for Configuring Automatic Migration of the JTA Transaction Recovery Service"

2.8 Roadmap for Setting Up a High Availability Topology

This section provides high level steps for configuring an example middleware topology with high availability, such as the topology that Section 1.5, "Understanding the Oracle Fusion Middleware Standard HA Topology" describes.

Table 2–5 describes the steps required to set up a high availability topology.

Table 2–5 Roadmap for Setting Up a High Availability Topology

Task	Description	Documentation
1. Install Real Application Clusters	Install Real Application Clusters	See *Oracle Real Application Clusters Administration and Deployment Guide*
2. Install middleware components	Install the application by following instructions in an application installation guide.	See *Oracle Fusion Middleware Installing and Configuring the Oracle Fusion Middleware Infrastructure* or the installation guide for your product
3. Install Oracle HTTP Server	Install Oracle HTTP Server in the same domain	See *Installing and Configuring Oracle HTTP Server*
4. Configure a load balancer	Configure a third-party load balancer that meets specific requirements, or Oracle HTTP Server/Oracle Traffic Director.	See Section 2.1, "Server Load Balancing in a High Availability Environment."
5. Scale out the topology (machine scale out)	Steps for scaling out a topology (machine scale-out) for all Fusion Middleware products that are a part of a Fusion Middleware WebLogic Server domain.	See Chapter 6, "Scaling Out a Topology (Machine Scale Out)"

Part II

Creating a High Availability Environment

Part II contains the following chapters:

- Chapter 3, "Using Shared Storage"
- Chapter 4, "Database Considerations"
- Chapter 5, "JMS and JTA High Availability"
- Chapter 6, "Scaling Out a Topology (Machine Scale Out)"

3 Using Shared Storage

This chapter provides basic recommendations for using shared storage in a high availability environment. It describes the benefits of placing artifacts in a common location that multiple hosts or servers share. This common location typically resides in a shared file system, which is mounted on each server with standard operating system protocols such as NFS and CIFS.

The following artifacts are typical candidates to place on a shared file system:

- **Product binaries**: All files and directories related to product executables, JAR files, and scripts that install during product installation.
- **Domain directory**: The directory containing the WebLogic Server domains and their configuration.
- **File-based persistence stores**: File-based persistence stores for JMS persistence and JTA transaction logs.

This chapter includes the following topics:

- Section 3.1, "Overview of Shared Storage"
- Section 3.2, "Shared Storage Prerequisites"
- Section 3.3, "Using Shared Storage for Binary (Oracle Home) Directories"
- Section 3.4, "Using Shared Storage for Domain Configuration Files"
- Section 3.5, "Shared Storage Requirements for JMS Stores and JTA Logs"
- Section 3.6, "Directory Structure and Configurations"

3.1 Overview of Shared Storage

Shared storage allows sharing of dynamic state and server configuration and simplifies administration, configuration, failover, and backup/recovery.

In a highly available environment, shared storage is required when using file based persistence stores (for JMS and JTA logs) and certain Oracle products. Shared storage is optional for product binaries and domain directories.

See Table 3–1 for additional information about shared storage.

Table 3–1 Shared Storage Topics

Topic/Task	For More Information
Structure and contents of an Oracle home	"Understanding the Oracle Fusion Middleware Infrastructure Directory Structure" in *Oracle Fusion Middleware Installing and Configuring the Oracle Fusion Middleware Infrastructure*
Saving JMS and JTA information in a file store	"Using the WebLogic Persistent Store" in *Administering Server Environments for Oracle WebLogic Server*. Includes the topic "High Availability for Persistent Stores"
	"Persistent Store High Availability" in *Administering JMS Resources for Oracle WebLogic Server*
	Default File Store Availability for JTA in *Administering Clusters for Oracle WebLogic Server*

3.2 Shared Storage Prerequisites

The following shared storage prerequisites apply only when you use file-based persistent stores:

- For proper recovery in the event of a failure, you must store both JMS and JTA transaction logs in a location that is accessible to all nodes that can resume operations after a Managed Server failure. This setup requires a shared storage location that multiple nodes can reference. See Section 3.6, "Directory Structure and Configurations" for the recommended directory structure.
- The shared storage can be a network-attached storage (NAS) or storage area network (SAN) device. For NFS-mounted systems, issues related to file locking and abrupt node failures have been detected. Check the *Oracle Fusion Middleware Release Notes* and with your storage vendor for the main recommended parameters for mount options.

  The following example command is based on a NAS device. Note that your options may be different from those in this example; see UNIX/Linux documentation for more information on the mount command and its options.

  ```
  mount nasfiler:/vol/vol1/u01/oracle /u01/oracle -t nfs -o
  rw,bg,hard,nointr,tcp,vers=3,timeo=300,rsize=32768,wsize=32768
  ```

- For maximum availability, Oracle recommends a highly available NAS or SAN device for shared storage. Shared storage devices that are not highly available can be a single point of failure. Check with your storage provider for options to achieve this.

For more information about saving JMS and JTA information in a file store, see "Using the WebLogic Persistent Store" in *Administering Server Environments for Oracle WebLogic Server*.

3.3 Using Shared Storage for Binary (Oracle Home) Directories

The following sections describe guidelines for using shared storage for your Oracle Fusion Middleware Oracle home directories:

- Section 3.3.1, "About the Binary (Oracle Home) Directories"
- Section 3.3.2, "About Using Redundant Binary (Oracle Home) Directories"

3.3.1 About the Binary (Oracle Home) Directories

When you install any Oracle Fusion Middleware product, you install the product binaries into an Oracle home. The binary files are read-only and do not change unless the Oracle home is patched or upgraded to a newer version.

In a typical production environment, the Oracle home files are saved in a separate location from the domain configuration files, which you create using the Oracle Fusion Middleware Configuration Wizard.

The Oracle home for an Oracle Fusion Middleware installation contains the binaries for Oracle WebLogic Server, the Oracle Fusion Middleware infrastructure files, and any Oracle Fusion Middleware product-specific directories.

Note: By default, the Configuration Wizard writes its logs to the logs directory in Oracle home. If you use a read-only Oracle home, you must specify the `-log` option to redirect logs to a different directory.

See Also: For more information about the structure and contents of an Oracle home, see "What are the Key Oracle Fusion Middleware Directories?" in *Oracle Fusion Middleware Understanding Oracle Fusion Middleware Concepts*.

3.3.2 About Using Redundant Binary (Oracle Home) Directories

For maximum availability, Oracle recommends using redundant binary installations on shared storage.

In this model, you install two identical Oracle homes for your Oracle Fusion Middleware software on two different shared volumes. You then mount one of the Oracle homes to one set of servers and the other Oracle home to the remaining servers. Each Oracle home has the same mount point, so the Oracle home always has the same path, regardless of which Oracle home the server is using.

If one Oracle home become corrupted or unavailable, only half your servers are affected. For additional protection, Oracle recommends that you disk mirror these volumes.

If separate volumes are not available on shared storage, Oracle recommends simulating separate volumes using different directories within the same volume and mounting these to the same mount location on the host side. Although this does not guarantee the protection that multiple volumes provide, it does allow protection from user deletions and individual file corruption.

3.4 Using Shared Storage for Domain Configuration Files

The following sections describe guidelines for using shared storage for the Oracle WebLogic Server domain configuration files you create when you configure your Oracle Fusion Middleware products in an enterprise deployment:

- Section 3.4.1, "About Oracle WebLogic Server Administration and Managed Server Domain Configuration Files"
- Section 3.4.2, "Shared Storage Considerations for Administration and Managed Server Domain Configuration Files"

3.4.1 About Oracle WebLogic Server Administration and Managed Server Domain Configuration Files

When you configure an Oracle Fusion Middleware product, you create or extend an Oracle WebLogic Server domain. Each Oracle WebLogic Server domain consists of a single Administration Server and one or more Managed Servers.

WebLogic uses a replication protocol to push persisted changes on the Administration Server to all Managed Servers. This gives redundancy to the Managed Servers so that you can start them without the Administration Server running. This mode is called Managed Server independence.

For more information about Oracle WebLogic Server domains, see *Understanding Domain Configuration for Oracle WebLogic Server*.

3.4.2 Shared Storage Considerations for Administration and Managed Server Domain Configuration Files

This section describes considerations for Administration Server and Managed Server configuration files.

Administration Server Configuration Directory

Oracle does not require that you store domain configuration files in shared storage. However, to support Administration Server recovery, you may choose to place the Administration Server configuration directory on shared storage and mount it on the host that is running the Administration Server. If that host fails, you can mount the directory on a different host and bring up the failed Administration Server on the other host.

Managed Server Configuration Files

Oracle recommends that you keep the Managed Server configuration files in local, or, host private, storage.

It is possible to keep Managed Server configuration files on shared storage. However, doing so can affect performance due to multiple servers concurrently accessing the same storage volume.

3.5 Shared Storage Requirements for JMS Stores and JTA Logs

When you use file-based persistence, it is mandatory to configure the JMS persistence stores and JTA transaction log directories to be in shared storage. For more information, see Section 5.4, "Considerations for Using File Persistence (WebLogic JMS)."

3.6 Directory Structure and Configurations

When you use shared storage, there are multiple ways to layout the storage elements. Oracle recommends the following best practices:

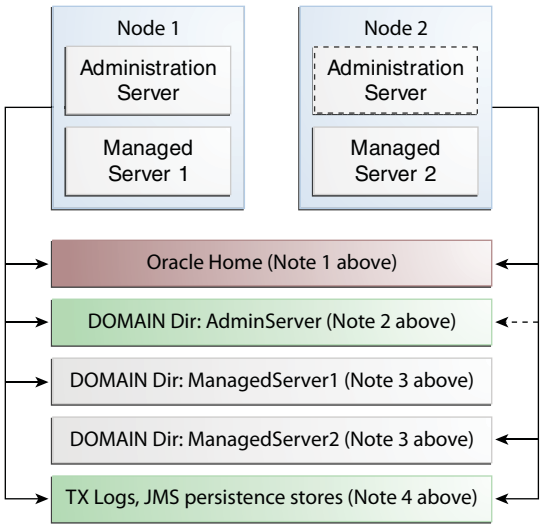
1. Place the product binaries (Oracle home) on shared storage and share it in a read-only mode by all servers.
2. Place the Administration Server domain configuration directory on shared storage to facilitate failing over the Administration server to a different host.
3. Place each Managed Server domain configuration directories on storage local to the corresponding host.

4. Place the JMS file stores and Transaction logs on shared storage if you use file-based persistence.

Figure 3–1 illustrates the directory structure.

Items that have arrows on both sides showing that they are connected to both nodes indicate the directories on shared storage. If Node 1 fails, the Administration Server domain directory is mounted on Node 2 (see the dotted arrow) to failover the Administration Server to Node 2.

Figure 3–1 Shared Storage Directory Structure



4 Database Considerations

This chapter describes what to consider as you configure database connections for Oracle Fusion Middleware in a high availability setup. It also describes the benefits of using Oracle Real Application Clusters (Oracle RAC), a commonly-deployed database high availability solution.

Most Fusion Middleware components use a database as the persistent store for their data. When you use an Oracle database, you can configure it in a variety of highly available configurations. (For more information on Oracle database options, see the *Oracle Database High Availability Overview*.)

This chapter includes the following topics:

- Section 4.1, "About Oracle Real Application Clusters"
- Section 4.2, "About RAC Database Connections and Failover"
- Section 4.3, "About Data Sources"
- Section 4.4, "Configuring Active GridLink Data Sources with Oracle RAC"
- Section 4.5, "Configuring Multi Data Sources"

4.1 About Oracle Real Application Clusters

A **cluster** comprises multiple interconnected computers or servers that appear as if they are one server to end users and applications. Oracle RAC enables you to cluster an Oracle database, providing a highly scalable and highly available database for Oracle Fusion Middleware.

All Oracle Fusion Middleware components deployed to Oracle WebLogic Server support Oracle RAC.

Every Oracle RAC instance in the cluster has equal access and authority, therefore, node and instance failure may affect performance, but doesn't result in downtime; the database service is available or can be made available on surviving server instances.

Table 4–1 outlines tasks and corresponding sources of information for setting up Oracle RAC.

Table 4–1 Roadmap for Setting up Oracle RAC

Task/Topic	More Information
About Oracle RAC	"Introduction to Oracle RAC" in the *Oracle Real Application Clusters Administration and Deployment Guide*
Installing Oracle RAC	*Oracle Real Application Clusters Administration and Deployment Guide*

Table 4–1 (Cont.) Roadmap for Setting up Oracle RAC

Task/Topic	More Information
Managing Oracle RAC	"Overview of Managing Oracle RAC Environments" in *Oracle Real Application Clusters Administration and Deployment Guide*
Configuring and tuning GridLink and multi data sources	*Oracle Fusion Middleware Configuring and Managing JDBC Data Sources for Oracle WebLogic Server*
Configuring Single Client Access Name (SCAN) URLs. (To specify the host and port for the TNS and ONS listeners in the WebLogic console.)	"SCAN Addresses" in the guide *Oracle Fusion Middleware Configuring and Managing JDBC Data Sources for Oracle WebLogic Server*.

4.2 About RAC Database Connections and Failover

To establish connection pools, Oracle Fusion Middleware supports Active GridLink data sources and multi data sources for the Oracle RAC back end for both XA and non-XA JDBC drivers. See Section 4.2.1, "About XA Transactions" for more information about XA transactions. These data sources also support load balancing across Oracle RAC nodes.

When an Oracle RAC node or instance fails, Oracle WebLogic Server or the Oracle Thin JDBC driver redirect session requests to another node in the cluster. There is no failover of existing connections. However, new connection requests from the application are managed using existing connections in the Oracle WebLogic pool or by new connections to the working Oracle RAC instance.

When the database is the transaction manager, in-flight transactions typically roll back.

When WebLogic Server is the transaction manager, in-flight transactions fail over; they are driven to completion or rolled back based on the transaction state when failure occurs.

4.2.1 About XA Transactions

XA transaction support enables multiple resources (such as databases, application servers, message queues, transactional caches) to be accessed within the same transaction. A *non-XA transaction* always involves just one resource.

An XA transaction involves a coordinating transaction manager with one or more databases, or other resources such as JMS, all involved in a single global transaction.

Java EE uses the terms **JTA transaction**, **XA transaction**, **user transaction**, and **global transaction** interchangeably to refer to a single global transaction. This type of transaction may include operations on multiple different XA- capable or non-XA resources and even different resource types. A JTA transaction is always associated with the current thread and may be passed from server to server as one application calls another. A common example of an XA transaction is one that includes both a WebLogic JMS operation and a JDBC (database) operation.

4.3 About Data Sources

A **data source** is an abstraction that application components use to obtain connections to a relational database. Specific connection information, such as the URL or user name and password, are set on a data source object as properties and do not need to be explicitly defined in an application's code. This abstraction allows applications to be

built in a portable manner, because the application is not tied to a specific back-end database. The database can change without affecting the application code.

Oracle provides Active GridLink data sources and multi data sources to support high availability, load balancing, and failover of database connections. Oracle recommends the following data source types depending on the Oracle RAC Database version you have:

- If you use Oracle RAC database version 11g Release 2 and later, use Active GridLink data sources.
- If you use an Oracle RAC database version earlier than 11g Release 2 or a non-Oracle database, use multi data sources.

> **Note:** Oracle recommends using the Active GridLink data sources with Oracle RAC database for maximum availability. For versions of Oracle RAC databases where Active GridLink data sources are not supported, Oracle recommends using multi data sources for high availability.

See the following topics for more information on these data source types:

- Section 4.3.1, "Active GridLink Data Sources"
- Section 4.3.2, "Multi Data Sources"

4.3.1 Active GridLink Data Sources

An **Active GridLink data source** provides connectivity between WebLogic Server and an Oracle database service, which may include multiple Oracle RAC clusters. An Active GridLink data source includes the features of generic data sources plus the following support for Oracle RAC:

- Uses the Oracle Notification Service (ONS) to respond to state changes in an Oracle RAC.
- Responds to Fast Application Notification (FAN) events to provide Fast Connection Failover (FCF), Runtime Connection Load-Balancing, and RAC instance graceful shutdown. FAN is a notification mechanism that Oracle RAC uses to quickly alert applications about configuration and workload.
- Provides Affinities (or XA Affinity) policies to ensure all database operations for a session are directed to the same instance of a RAC cluster for optimal performance.
- SCAN Addresses
- Secure Communication Using Oracle Wallet

See "Using Active GridLink Data Sources" in the *Oracle Fusion Middleware Configuring and Managing JDBC Data Sources for Oracle WebLogic Server* guide for more information on the following topics:

- What is an Active GridLink Data Source
- Using Socket Direct Protocol
- Configuring Connection Pool Features
- Configuring Oracle Parameters
- Configuring an ONS Client

- Tuning Active GridLink Data Source Connection Pools
- Monitoring GridLink JDBC Resources

4.3.2 Multi Data Sources

A **multi data source** is an abstraction around a group of data sources that provides load balancing or failover processing at the time of connection requests, between the data sources associated with the multi data source. Multi data sources support load balancing for both XA and non-XA data sources.

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

Multi data sources are bound to the JNDI tree or local application context just like regular data sources. Applications look up a multi data source on the JNDI tree or in the local application context (`java:comp/env`) just as they do for data sources, and then request a database connection. The multi data source determines which data source to use to satisfy the request depending on the algorithm selected in the multi data source configuration: load balancing or failover.

See Also: For more information about configuring Multi Data Sources with Oracle RAC, see "Using Multi Data Sources with Oracle RAC" in the guide *Oracle Fusion Middleware Configuring and Managing JDBC Data Sources for Oracle WebLogic Server*.

4.4 Configuring Active GridLink Data Sources with Oracle RAC

How you configure an Active GridLink data source depends on the Oracle component that you are working with and the domain you are creating.

This section describes how to configure component data sources as Active GridLink data sources for a RAC database during domain creation.

This topic includes the following sections:

- Section 4.4.1, "Requirements"
- Section 4.4.2, "Configuring Component Data Sources as Active GridLink Data Sources"
- Section 4.4.3, "Using Single Client Access Name (SCAN) Addresses for Hosts and Ports"

See Also: To create and configure Active GridLink data sources, see Using Active GridLink Data Sources in *Oracle Fusion Middleware Configuring and Managing JDBC Data Sources for Oracle WebLogic Server*.

4.4.1 Requirements

Verify that your system meets the following requirements before you configure component data sources as Active GridLink data sources to use with an Oracle RAC database:

- You are using Oracle RAC database version 11g Release 2 or later.

- You have run RCU to create component schemas.
- You are using the Configuration Wizard to create or configure a domain and have arrived at the JDBC Component Schema screen where you select **Component Datasources**. Before you arrive at the JDBC Component Schema screen, you must select the option **Configure Manually** in the Database Configuration Type screen.

4.4.2 Configuring Component Data Sources as Active GridLink Data Sources

To configure component data sources as Active GridLink data sources:

1. In the JDBC Component Schema screen, select one or more component schemas to configure GridLink data sources for.
2. Select **Convert to GridLink** then select **Next**.
3. In the GridLink Oracle RAC Component Schema screen, select one of the GridLink JDBC drivers.
4. In the **Service Name** field, enter the service name of the database using lowercase characters. For example, `mydb.example.com`.
5. In the **Schema Owner** field, enter the name of the database schema owner for the corresponding component.
6. In the **Schema Password** field, enter the password for the database schema owner.
7. In the **Service Listener**, **Port**, and **Protocol** field, enter the SCAN address and port for the RAC database being used. The protocol for Ethernet is TCP; for Infiniband it is SDP. Click **Add** to enter multiple listener addresses.

   You can identify the SCAN address by querying the appropriate parameter in the database using the TCP protocol:

   ```
   show parameter remote_listener

   NAME                 TYPE                 VALUE
   ---------------------------------------------------------------------
   remote_listener      string               db-scan.example.com:1521
   ```

   You can also identify the SCAN address by using the `srvctl config scan` command. Use the command `srvctl config scan_listener` to identify the SCAN listener port.
8. Select **Enable FAN** to receive and process FAN events. Enter one or more ONS daemon listen addresses and port information. Select **Add** to enter more entries.

   Note: Verify that the ONS daemon listen address(es) that you enter is valid. The address is not validated during the domain creation process.

   For the ONS host address, you can use the SCAN address for the Oracle RAC database and the ONS remote port as reported by the database:

   ```
   srvctl config nodeapps -s

   ONS exists: Local port 6100, remote port 6200, EM port 2016
   ```
9. Select **Enable SSL** for SSL communication with ONS. Enter the Wallet File, which has the SSL certificates, and the Wallet Password.

10. Select **Next**. Verify that all connections are successful.

Note: See "Modifying the mdsDS Data Source URL" to use an Active GridLink data source with a customer-provided ADF application.

See Also: For more information, see:

- "JDBC Component Schema" in *Oracle Fusion Middleware Creating WebLogic Domains Using the Configuration Wizard* for information about the JDBC Component Schema screen.
- "GridLink Oracle RAC Component Schema" in *Oracle Fusion Middleware Creating WebLogic Domains Using the Configuration Wizard* for information about configuring component schemas.
- "Using Active GridLink Data Sources" in *Administering JDBC Data Sources for Oracle WebLogic Server* for information on GridLink RAC data sources.

4.4.3 Using Single Client Access Name (SCAN) Addresses for Hosts and Ports

Oracle recommends that you use Oracle Single Client Access Name (SCAN) addresses to specify the host and port for both the TNS listener and the ONS listener in the WebLogic console. You do not need to update an Active GridLink data source containing SCAN addresses if you add or remove Oracle RAC nodes. Contact your network administrator for appropriately configured SCAN URLs for your environment. See SCAN Addresses in the *Oracle Fusion Middleware Configuring and Managing JDBC Data Sources for Oracle WebLogic Server* guide.

4.5 Configuring Multi Data Sources

You can configure multi data sources using the following:

- Oracle Fusion Middleware Configuration Wizard during WebLogic Server domain creation
- Oracle WebLogic Server Administration Console
- WLST Commands

This section includes the following topics:

- Section 4.5.1, "Configuring Multi Data Sources with Oracle RAC"
- Section 4.5.2, "Configuring Multi Data Sources for MDS Repositories"

4.5.1 Configuring Multi Data Sources with Oracle RAC

This section describes the requirements and procedure to configure multi data sources with Oracle RAC.

This section includes the following topics:

- Section 4.5.1.1, "Requirements"
- Section 4.5.1.2, "Configuring Component Data Sources as Multi Data Sources"
- Section 4.5.1.3, "Modifying or Creating Multi Data Sources After Initial Configuration"

4.5.1.1 Requirements

Verify that your system meets the following requirements before you configure component data sources as multi data sources to use with an Oracle RAC database:

- You are using an Oracle RAC database.
- You have run RCU to create component schemas.
- You are using the Configuration Wizard to create or configure a domain and have arrived at the JDBC Component Schema Screen where you select Component Schemas.

4.5.1.2 Configuring Component Data Sources as Multi Data Sources

To configure component data sources as multi data sources:

1. In the Component Datasources screen, select one or more component schemas to configure Active GridLink data sources for.
2. Select **Convert to RAC multi data source** then select **Next**.
3. In the Oracle RAC Multi Data Source Component Schema screen, the JDBC driver **Oracle's Driver (Thin) for RAC Service-Instance connections; Versions:10 and later**.
4. In the **Service Name** field, enter the database service name enter in lowercase, for example, `mydb.example.com`.
5. In the **Schema Owner** field, enter the username of the database schema owner for the corresponding component. Typically this name is *RCUprefix_component*.
6. In the **Schema Password** field, enter the password for the database schema owner.
7. In the **Host Name**, **Instance Name**, and **Port** field, enter the RAC node hostname, database instance name, and port. Click **Add** to enter multiple listener addresses.
8. Click **Next**.Verify that all connections are successful.

4.5.1.3 Modifying or Creating Multi Data Sources After Initial Configuration

The multi data sources have constituent data sources for each RAC instance providing the database service. Oracle recommends that you add an additional data source to the multi data source on the Fusion Middleware tier when you add an additional instance to the RAC back end.

When you migrate a database from a non-RAC to a RAC database, you must create an equivalent, new multi data source for each data source that is affected. The multi data source that you create must have constituent data sources for each RAC instance. The data source property values must be identical to the original single instance data source for the properties in Section 4.5.1. For example, if the single instance data source driver is `oracle.jdbc.xa.client.OracleXADataSource`, it must be `oracle.jdbc.xa.client.OracleXADataSource` for each constituent data source of the new multi data source.

For multi data sources that you create manually or modify after initial configuration, Oracle strongly recommends specific XA and non-XA data source property values for optimal high availability. Make changes only after careful consideration and testing if your environment requires that you do so.

The following tables describe XA and non-XA data source property values that Oracle recommends:

- Table 4–2, " Recommended Multi Data Source Configuration"

- Table 4–3, " XA Data Source Configuration"
- Table 4–4, " Non-XA Data Source Configuration"

Table 4–2 Recommended Multi Data Source Configuration

Property Name	Recommended Value
test-frequency-seconds	5
algorithm-type	Load-Balancing

For individual data sources, Oracle recommends the following for high availability environments. Oracle recommends that you set any other parameters according to application requirements.

Table 4–3 XA Data Source Configuration

Property Name	Recommended Value
Driver	oracle.jdbc.xa.client.OracleXADataSource
Property command	<property> <name>oracle.net.CONNECT_TIMEOUT</name> <value>10000</value> </property>
initial-capacity	0
connection-creation-retry-frequency-seconds	10
test-frequency-seconds	300
test-connections-on-reserve	true
test-table-name	SQL SELECT 1 FROM DUAL
seconds-to-trust-an-idle-pool-connection	0
global-transactions-protocol	TwoPhaseCommit
keep-xa-conn-till-tx-complete	true
xa-retry-duration-seconds	300
xa-retry-interval-seconds	60

Troubleshooting Warning Messages (Increasing Transaction Timeout for XA Data Sources)

If you see WARNING messages in the server logs that include the following exception, this message may indicate that the XA timeout value you have in your setup must be increased.

```
[ javax.transaction.SystemException: Timeout during commit processing
```

You can increase XA timeout for individual data sources when these warnings appear.

To increase the transaction timeout for the XA Data Sources setting, use the Administration Console:

1. Access the data source configuration.
2. Select the **Transaction** tab.
3. Set the XA Transaction Timeout to a larger value, for example, **300**.

4. Select the **Set XA Transaction Timeout** checkbox. You *must* select this checkbox for the new XA transaction timeout value to take effect.
5. Click **Save**.

Repeat this configuration for all individual data sources of an XA multi data source.

Table 4–4 Non-XA Data Source Configuration

Property Name	Recommended Value
Driver	oracle.jdbc.OracleDriver
Property to set	<property> <name>oracle.net.CONNECT_TIMEOUT</name> <value>10000</value> </property>
initial-capacity	0
connection-creation-retry-frequency-seconds	10
test-frequency-seconds	300
test-connections-on-reserve	true
test-table-name	SQL SELECT 1 FROM DUAL
seconds-to-trust-an-idle-pool-connection	0
global-transactions-protocol	None

4.5.2 Configuring Multi Data Sources for MDS Repositories

You can configure applications that use an MDS database-based repository for high availability Oracle database access. With this configuration, failure detection, recovery, and retry by MDS (and by the WebLogic infrastructure) result in application read-only MDS operations being protected from Oracle RAC database planned and unplanned downtimes.

Multi data sources are exposed as MDS repositories in the Fusion Middleware Control navigation tree. You can select these multi data sources when you customize the application deployment and use them with MDS WLST commands.

- Configuring an application to retry read-only operations

  To configure an application to retry the connection, you can configure the `RetryConnection` attribute of the application's MDS AppConfig MBean. See the *Oracle Fusion Middleware Administrator's Guide* for more information.

- Registering an MDS multi data source

  In addition to the steps in Section 4.5.1, "Configuring Multi Data Sources with Oracle RAC," consider the following:

  – The child data sources that constitute a multi data source used for an MDS repository must be configured as non-XA data sources.

  – The multi data source's name must have the prefix `mds-`. This ensures that the multi data source is recognized as an MDS repository that can be used for MDS management functionality through Fusion Middleware Control, WLST, and JDeveloper.

Note: When an MDS data source is added as a child of a multi data source, this data source is no longer exposed as an MDS repository. For example, it does not appear under the Metadata Repositories folder in the Fusion Middleware Control navigation tree, you cannot perform MDS repository operations on it, and it does not appear in the list of selectable repositories during deployment.

- Converting a data source to a multi data source

  There are two things to consider when you convert a data source to a multi data source to verify that the application is configured correctly:

  – To create a new multi data source with a new, unique name, redeploy the application and select this new multi data source as the MDS repository during deployment plan customization.

  – To avoid redeploying the application, you can delete the data source and recreate the new multi data source using the same name and jndi-name attributes.

Note: See "Modifying the mdsDS Data Source URL" to use a multi data source with a customer-provided ADF application.

---

5 JMS and JTA High Availability

This chapter describes Java Message Service (JMS) and Java Transaction API (JTA) high availability.

This chapter includes the following topics:

- Section 5.1, "About JMS and JTA Services for High Availability"
- Section 5.2, "Configuring JMS and JTA Services for High Availability"
- Section 5.3, "User-Preferred Servers and Candidate Servers"
- Section 5.4, "Considerations for Using File Persistence (WebLogic JMS)"
- Section 5.5, "Configuring Schemas for Transactional Recovery Privileges"

See Also: For more information on working with JMS or JTA, see one of the following topics:

- "Configuring WebLogic JMS Clustering" in the guide *Oracle Fusion Middleware Administering JMS Resources for Oracle WebLogic Server*
- "Interoperating with Oracle AQ JMS" in the guide *Oracle Fusion Middleware Administering JMS Resources for Oracle WebLogic Server*
- "Configuring JTA" in the guide *Developing JTA Applications for Oracle WebLogic Server*.
- "Domains:Configuration:JTA" and "Cluster:Configuration:JTA" in the *Administration Console Online Help*

5.1 About JMS and JTA Services for High Availability

Java Message Service (JMS) is an application program interface (API) that supports the formal communication known as *messaging* between computers in a network.

Java Transaction API (JTA) specifies standard Java interfaces between a transaction manager and the parties involved in a distributed transaction system: the resource manager, the application server, and the transactional applications.

You can configure JMS and JTA services for high availability by using a **migratable target**, a special target that can migrate from one server in a cluster to another. A migratable target provides a way to group migratable services that should move together. When a migratable target migrates, all services the target hosts also migrate.

5.2 Configuring JMS and JTA Services for High Availability

To configure a migratable JMS service for migration, you must deploy it to a migratable target. The migratable target specifies a set of servers that can host a target, and can specify a user-preferred host for the services and an ordered list of candidate backup servers if the preferred server fails. Only one server can host the migratable target at any one time.

After you configure a service to use a migratable target, the service is independent from the server member that is currently hosting it. For example, if a JMS server with a deployed JMS queue is configured to use a migratable target then the queue is independent of when a specific server member is available. That is, the queue is always available when the migratable target is hosted by any server in the cluster.

You can manually migrate pinned migratable services from one server instance to another in the cluster if a server fails or as part of regularly scheduled maintenance. If you do not configure a migratable target in the cluster, migratable services can migrate to any WebLogic Server instance in the cluster.

See Also: For more information on administering JMS, see the guide *Oracle Fusion Middleware Administering JMS Resources for Oracle WebLogic Server*, which includes the following topics:

- "High Availability Best Practices"
- "Interoperating with Oracle AQ JMS"

5.3 User-Preferred Servers and Candidate Servers

When you deploy a JMS service to the migratable target, you can select a user-preferred server target to host the service. When configuring a migratable target, you can also specify constrained candidate servers (CCS) that can host the service if the user-preferred server fails. If the migratable target does not specify a constrained candidate server, you can migrate the JMS server to any available server in the cluster.

WebLogic Server enables you to create separate migratable targets for JMS services. This allows you to always keep each service running on a different server in the cluster, if necessary. Conversely, you can configure the same selection of servers as the constrained candidate servers for both JTA and JMS, to ensure that the services remain co-located on the same server in the cluster.

See Also: For more information, see the following topics in the *Administration Console Online Help*:

- "Configure migratable targets for JMS-related services"
- "Configure migratable targets for the JTA Transaction Recovery Service"

5.4 Considerations for Using File Persistence (WebLogic JMS)

You can configure JMS messages and JTA logs to be stored on the file system. For high availability, you must use a shared file system. See Chapter 3, "Using Shared Storage" for information on what to consider when using a shared file system to store these artifacts.

See Also: For more information see "WebLogic JMS Architecture and Environment" in the guide *Administering JMS Resources for Oracle WebLogic Server.*

5.4.1 Considerations for Using File Stores on NFS

If you store JMS messages and transaction logs on an NFS-mounted directory, Oracle strongly recommends that you verify the behavior of a server restart after an abrupt machine failure. Depending on the NFS implementation, different issues can arise after a failover/restart.

To verify server restart behavior, abruptly shut down the node that hosts WebLogic servers while the servers are running.

- If you configured the server for server migration, it should start automatically in failover mode after the failover period.
- If you did not configure the server for server migration, you can manually restart the WebLogic Server on the same host after the node completely reboots.

If Oracle WebLogic Server does not restart after abrupt machine failure, verify whether or not it is due to an I/O exception that is similar to the following by reviewing the server log files:

```
<MMM dd, yyyy hh:mm:ss a z> <Error> <Store> <BEA-280061> <The persistent
store "_WLS_server_1" could not be deployed:
weblogic.store.PersistentStoreException: java.io.IOException:
[Store:280021]There was an error while opening the file store file
"_WLS_SERVER_1000000.DAT"
weblogic.store.PersistentStoreException: java.io.IOException:
[Store:280021]There was an error while opening the file store file
"_WLS_SERVER_1000000.DAT"
        at weblogic.store.io.file.Heap.open(Heap.java:168)
        at weblogic.store.io.file.FileStoreIO.open(FileStoreIO.java:88)
...
java.io.IOException: Error from fcntl() for file locking, Resource
temporarily unavailable, errno=11
```

This error occurs when the NFSv3 system does not release locks on the file stores. WebLogic Server maintains locks on files that store JMS data and transaction logs to prevent data corruption that can occur if you accidentally start two instances of the same Managed Server. Because the NFSv3 storage device doesn't track lock owners, NFS holds the lock indefinitely if a lock owner fails. As a result, after abrupt machine failure followed by a restart, subsequent attempts by WebLogic Server to acquire locks may fail.

How you resolve this error depends on your NFS environment: (See *Oracle Fusion Middleware Release Notes* for updates on this topic.)

- **For NFSv4 environments**, you can set a tuning parameter on the NAS server to release locks within the approximate time required to complete server migration; you do not need to follow the procedures in this section. See your storage vendor's documentation for information on locking files stored in NFS-mounted directories on the storage device, and test the results.
- **For NFSv3 environments**, the following sections describe how to disable WebLogic file locking mechanisms for: the default file store, a custom file store, a JMS paging file store, a diagnostics file store.

WARNING: NFSv3 file locking prevents severe file corruptions that occur if more than one Managed Server writes to the same file store at any point in time.

If you disable NFSv3 file locking, you must implement administrative procedures / policies to ensure that only one Managed Server writes to a specific file store. Corruption can occur with two Managed Servers in the same cluster or different clusters, on the same node or different nodes, or on the same domain or different domains.

Your policies could include: never copy a domain, never force a unique naming scheme of WLS-configured objects (servers, stores), each domain must have its own storage directory, no two domains can have a store with the same name that references the same directory.

If you configure a Managed Server using a file store for server migration, always configure the database- based leasing option. This option enforces additional locking mechanisms using database tables and prevents automated restart of more than one instance of a particular Managed Server.

Disabling File Locking for the Default File Store

To disable file locking for the default file store using the Administration Console:

1. If necessary, click **Lock & Edit** in the Change Center (upper left corner) of the Administration Console to get an Edit lock for the domain.
2. In the **Domain Structure** tree, expand the **Environment** node and select **Servers**.
3. In the **Summary of Servers** list, select the server you want to modify.
4. Select the **Configuration > Services** tab.
5. Scroll down to the **Default Store** section and click **Advanced**.
6. Scroll down and deselect the **Enable File Locking** check box.
7. Click **Save**. If necessary, click **Activate Changes** in the Change Center.
8. **Restart** the server you modified for the changes to take effect.

The resulting `config.xml` entry looks like the following:

```
<server>
  <name>examplesServer</name>
  ...
  <default-file-store>
    <synchronous-write-policy>Direct-Write</synchronous-write-policy>
    <io-buffer-size>-1</io-buffer-size>
    <max-file-size>1342177280</max-file-size>
    <block-size>-1</block-size>
    <initial-size>0</initial-size>
    <file-locking-enabled>false</file-locking-enabled>
  </default-file-store>
</server>
```

Disabling File Locking for a Custom File Store

To disable file locking for a custom file store using the Administration Console:

1. If necessary, click **Lock & Edit** in the Change Center (upper left corner) of the Administration Console to get an Edit lock for the domain.
2. In the **Domain Structure** tree, expand the **Services** node and select **Persistent Stores**.
3. In the **Summary of Persistent Stores** list, select the custom file store you want to modify.
4. On the **Configuration** tab for the custom file store, click **Advanced**.
5. Scroll down and deselect the **Enable File Locking** check box.
6. Click **Save**. If necessary, click **Activate Changes** in the Change Center.
7. If the custom file store was in use, you must restart the server for the changes to take effect.

The resulting `config.xml` entry looks like the following:

```
<file-store>
  <name>CustomFileStore-0</name>
  <directory>C:\custom-file-store</directory>
  <synchronous-write-policy>Direct-Write</synchronous-write-policy>
  <io-buffer-size>-1</io-buffer-size>
  <max-file-size>1342177280</max-file-size>
  <block-size>-1</block-size>
  <initial-size>0</initial-size>
  <file-locking-enabled>false</file-locking-enabled>
  <target>examplesServer</target>
</file-store>
```

Disabling File Locking for a JMS Paging File Store

To disable file locking for a JMS paging file store using the Administration Console:

1. If necessary, click **Lock & Edit** in the Change Center (upper left corner) of the Administration Console to get an Edit lock for the domain.
2. In the **Domain Structure** tree, expand the **Services** node, expand the **Messaging** node, and select **JMS Servers**.
3. In the **Summary of JMS Servers** list, select the JMS server you want to modify.
4. On the **Configuration > General** tab for the JMS Server, scroll down and deselect the **Paging File Locking Enabled** check box.
5. Click **Save**. If necessary, click **Activate Changes** in the Change Center.
6. **Restart** the server you modified for the changes to take effect.

The resulting `config.xml` file entry will look like the following:

```
<jms-server>
  <name>examplesJMSServer</name>
  <target>examplesServer</target>
  <persistent-store>exampleJDBCStore</persistent-store>
  ...
  <paging-file-locking-enabled>false</paging-file-locking-enabled>
  ...
</jms-server>
```

Disabling File Locking for a Diagnostics File Store

To disable file locking for a Diagnostics file store using the Administration Console:

1. If necessary, click **Lock & Edit** in the Change Center (upper left corner) of the Administration Console to get an Edit lock for the domain.
2. In the **Domain Structure** tree, expand the **Diagnostics** node and select **Archives**.
3. In the **Summary of Diagnostic Archives** list, select the server name of the archive that you want to modify.
4. On the **Settings for [server_name]** page, deselect the **Diagnostic Store File Locking Enabled** check box.
5. Click **Save**. If necessary, click **Activate Changes** in the Change Center.
6. **Restart** the server you modified for the changes to take effect.

The resulting `config.xml` file will look like this:

```
  <server>
    <name>examplesServer</name>
    ...
    <server-diagnostic-config>
      <diagnostic-store-dir>data/store/diagnostics</diagnostic-store-dir>
      <diagnostic-store-file-locking-enabled>false</diagnostic-store-file-locking-
enabled>

<diagnostic-data-archive-type>FileStoreArchive</diagnostic-data-archive-type>
      <data-retirement-enabled>true</data-retirement-enabled>
      <preferred-store-size-limit>100</preferred-store-size-limit>
      <store-size-check-period>1</store-size-check-period>
    </server-diagnostic-config>
  </server>
```

> **See Also:** For more information, see one of the following topics:
>
> - "Configure JMS Servers and Persistent Stores" in *Oracle Fusion Middleware Administering JMS Resources for Oracle WebLogic Server*.
> - "Using the WebLogic Persistent Store" in *Oracle Fusion Middleware Configuring Server Environments for Oracle WebLogic Server*.

5.5 Configuring Schemas for Transactional Recovery Privileges

You need the appropriate database privileges to enable the WebLogic Server transaction manager to:

- Query for transaction state information
- Issue the appropriate commands, such as commit and rollback, during recovery of in-flight transactions after a WebLogic Server container failure.

To configure the schemas for transactional recovery privileges:

1. Log on to SQL*Plus as a user with sysdba privileges. For example:

   ```
   sqlplus "/ as sysdba"
   ```

2. Grant select on `sys.dba_pending_transactions` to the `appropriate_user`.
3. Grant force any transaction to the `appropriate_user`.

6

Scaling Out a Topology (Machine Scale Out)

This chapter describes the steps for scaling out a topology (machine scale-out) for all Fusion Middleware products that are a part of a Fusion Middleware WebLogic Server domain. To enable high availability, it is important to provide failover capabilities to another host computer. When you do so, your environment can continue to serve the consumers of your deployed applications if one computer goes down.

This chapter includes the following topics:

- Section 6.1, "About Machine Scale Out"
- Section 6.2, "Roadmap for Scaling Out Your Topology"
- Section 6.3, "Optional Scale Out Procedure"
- Section 6.4, "About Scale Out Prerequisites"
- Section 6.5, "Resource Requirements"
- Section 6.6, "Creating a New Machine"
- Section 6.7, "Packing the Domain on APPHOST1"
- Section 6.8, "Preparing the New Machine"
- Section 6.9, "Running Unpack to Transfer the Template"
- Section 6.10, "Starting the Node Manager"
- Section 6.11, "Starting the Managed Servers"
- Section 6.12, "Verifying Machine Scale Out"

6.1 About Machine Scale Out

Scalability is the ability of a piece of hardware or software or a network to "expand" or "shrink" to meet future needs and circumstances. A scalable system is one that can handle increasing numbers of requests without adversely affecting response time and throughput.

Machine scale-out is the process of moving a server, one of many on one machine, to another machine for high availability. Machine scale out is different from Managed Server **scale up**, which is the process of adding a new Managed Server to a machine that already has one or more Managed Servers running on it. For more information on scaling up your environment, see "Scaling Up Your Environment" in the guide *Oracle Fusion Middleware Administering Oracle Fusion Middleware*.

6.2 Roadmap for Scaling Out Your Topology

The following table describes the typical steps you must take to scale out a topology.

Table 6–1 Roadmap for Scaling Out Your Topology

Task	Description	More Information
Product is ready for scale out	Product is installed, configured, and a cluster of Managed Servers is available; your product is in a standard installation topology	Section 6.4, "About Scale Out Prerequisites"
Verify that you meet resource requirements	You must verify that your environment meets certain requirements	Section 6.5, "Resource Requirements"
Create a new machine and assign servers to it	Use the Administration Console to create a new machine and add Managed Servers to it.	Section 6.6, "Creating a New Machine"
Run the pack command	Pack up the domain directory	Section 6.7, "Packing the Domain on APPHOST1"
Prepare the new machine	Install the same software that you installed on the first machine	Section 6.8, "Preparing the New Machine"
Run the unpack command	Create a Managed Server template.	Section 6.9, "Running Unpack to Transfer the Template"
Start the server	Starts the Managed Server on the new machine	Section 6.11, "Starting the Managed Servers"
Verify the topology	Test the new setup	Section 6.12, "Verifying Machine Scale Out"

Note: In this section, *APPHOST* refers to a physical host computer, and *machine* refers to the WebLogic Server machine definition describing that host. See "Understanding the Oracle Fusion Middleware Infrastructure Standard Installation Topology" in the guide *Installing and Configuring the Oracle Fusion Middleware Infrastructure*.

6.3 Optional Scale Out Procedure

Following the standard installation topology results in multiple Managed Servers assigned to a single host computer

This approach represents the most flexible way to create and scale out a domain topology so it can meet a variety of changing requirements. It allows you to 1) create and validate a single-host domain, which is targeted to a single machine on a single host computer, and then 2) "retarget" the Managed Servers to additional machines, as additional computing resources are required. An additional benefit of this approach is that it facilitates troubleshooting; you can validate the basic domain then perform and troubleshoot scale up and scale out steps at a later time.

However, if you know ahead of time what your target topology is, you can create machines during additional domain creation then simply perform the pack and unpack steps.

If you have already assigned Managed Servers to their target machines, either in the initial installation process or through an online administrative operation, simply skip Section 6.6, "Creating a New Machine" as you progress through the roadmap (Table 6–1, " Roadmap for Scaling Out Your Topology").

See the product installation guides, such as *Installing and Configuring the Oracle Fusion Middleware Infrastructure*, for more information on machine mapping.

6.4 About Scale Out Prerequisites

Before you start the scale out process, you must have a **standard installation topology** set up for your product. The standard installation topology serves as the starting point for scale out. If you followed the steps in your product installation guide, you should have a standard installation topology. For an example, see the standard installation topology that the topic "Understanding the Oracle Fusion Middleware Infrastructure Standard Installation Topology" describes in the guide *Oracle Fusion Middleware Installing and Configuring the Oracle Fusion Middleware Infrastructure.*

See Also: For more information on the standard installation topology, see your product's installation guide or "About the Standard Installation Topology" in the guide *Planning an Installation of Oracle Fusion Middleware*.

6.5 Resource Requirements

Before you scale out the topology, verify that your environment meets these requirements:

- At least one machine running multiple Managed Servers configured with a product. This is the result of following your product installation guide or administration guide to add additional servers.
- A host computer in addition to your starting host computer.
- Each host computer can access the Oracle home that contains the product binaries by one of the following means:
  - Shared disk with binaries from the original installation
  - Dedicated disk with a new installation (matches the original installation)
  - Dedicated disk with a clone of the binaries from the original installation

  See Chapter 3, "Using Shared Storage" for more information.
- Sufficient storage available for the domain directory.
- Access to the same Oracle or third-party database used for the original installation.
- A shared disk for JMS and transaction log files (required when using a file persistence profile).

6.6 Creating a New Machine

A **machine** is the logical representation of the computer that hosts one or more WebLogic Server instances (servers). In a WebLogic domain, the machine definitions identify physical units of hardware and are associated with the WebLogic Server instances that they host.

Follow these steps to create a new machine:

- Section 6.6.1, "Shutting Down the Managed Server"
- Section 6.6.2, "Creating a New Machine"
- Section 6.6.3, "Assigning Managed Servers to the New Machine"

6.6.1 Shutting Down the Managed Server

The server must be in a shutdown state before moving to a new machine. If the server is up and running, see the topic "Shut down a server instance" in the *Administration Console Online Help* to shut down the Managed Server that the new machine will host.

6.6.2 Creating a New Machine

This topic describes how to create a new machine using the Administration Console. To create a new machine using WLST commands see "Creating a New Machine for Certain Components" in the guide *Oracle Fusion Middleware Administering Oracle Fusion Middleware*.

> **Note:** The machine you create in this procedure must have a listen address on a specific network interface, not just a local host.

To create a new machine in the domain using the Administration Console, perform the following steps:

1. If the domain Administration Server is not running, you must start it. Go to the DOMAIN_HOME/bin directory and run:

   ```
   ./startWeblogic.sh
   ```

2. After the Administration Server is up and running, access the WebLogic Server Administration Console. Open a web browser and enter the URL:

   `http://hostname:port/console`

   On the Welcome screen, log in.

3. In the Change Center of the Administration Console, click **Lock & Edit**.

   > **Note:** For production domains, Oracle recommends creating the domain in "Production" mode, which enables Change Center. If Production mode is *not* enabled, Change Center steps are not required.

4. Under Domain Structure, expand Environment then click **Machines**.
5. Above the Machines table (above Summary), click **New**.
6. In the Create a New Machine screen, enter a name for the machine (such as **machine_2**). Select the **Machine OS** using the drop-down list then click **Next**.

7. On the next screen, for **Type:**, use the drop-down list to select **Plain**.

   For the Node Manager **Listen Address**, enter the IP address or host name of the host computer that will represent this machine. Both machines appear in the Machines table.

8. Click **Finish**.

9. In the Change Center, click **Activate Changes**.

   The message **"All changes have been activated. No restarts are necessary.**" appears. This message indicates that you have a new machine.

6.6.3 Assigning Managed Servers to the New Machine

To add Managed Servers to the newly-created machine, use the Administration Console to perform the following steps:

1. In the Change Center, click **Lock & Edit**.

2. In the Machines table, click the machine **machine_1**.

3. Click the machine name.

4. Under the Settings for **machine_1**, click the Configuration tab and then the Servers subtab.

5. Above the Servers table, click **Add**.

6. On the Add a Server to Machine screen, enter the **Server Name** and the **Server Listen Port** in the fields (required).

   Use the Select a server drop-down list to choose **machine_1** and then click **Finish**.

7. Under Domain Structure, click **Machines**.

   In the Machines table, click the machine **machine_2**.

8. Under the Settings for **machine_2**, click the Configuration tab and then the Servers tab. Above the Servers tab, click **Add**.

   On the **Add a Server to Machine** screen, select the button **Select an existing server**, **and associate it with this machine**.

   Use the Select a server drop-down list to choose **machine_2** then select **Finish**.

   The message **Server created successfully** appears.

9. To complete the changes, go back to the Change Center. Click **Activate Changes**. The message **All changes have been activated. No restarts are necessary.** appears.

   To see a summary, under Domain Structure, under Environment, click **Servers**. The Servers table on the right shows all servers in the domain and their machine assignments.

6.7 Packing the Domain on APPHOST1

You create a Managed Server template by running the `pack` command on the WebLogic domain.

Note: The Administration Server should be running on APPHOST1 when you go through the pack and unpack steps.

Run the `pack` command on **APPHOST1** to create a template pack. You will unpack the template file on **APPHOST2** later in the scale out process; Section 6.9, "Running Unpack to Transfer the Template" describes the unpack procedure.

For example:

```
ORACLE_COMMON/common/bin/pack.sh \
     -domain=DOMAIN_HOME \
     -template=dir/domain_name.jar \
     -managed=true \
     -template_name="DOMAIN"
```

In the preceding example:

- Replace *DOMAIN_HOME* with the full path to the domain home directory.
- Replace *dir* with the full path to a well-known directory where you will create the new template file.
- Replace *domain_name* in the JAR file name with the domain name. This is the name of the template file that you are creating with the pack command. For example: `mydomain.jar`.

> **See Also:** See "pack and unpack Command Reference" in *Creating WebLogic Domains and Domain Templates* for more information about creating a Managed Server template.

6.8 Preparing the New Machine

To prepare the new machine, **machine_2**, verify that **APPHOST2** has access to the shared disk where the Oracle home is installed or install the same software that you installed on **machine_1**.

For example, if you are scaling out an Oracle Fusion Middleware Infrastructure domain, verify that **APPHOST2** can access the Infrastructure Oracle home.

> **Note:** If you use shared storage, you can reuse the same installation location.

> **Note:** If you are performing a new installation or reusing the binaries by means of shared storage, the path location of the new files must match the original machine's path location exactly.

6.9 Running Unpack to Transfer the Template

To unpack the template and transfer the *domain_name*.jar file from **APPHOST1** to **APPHOST2**, run the unpack command:

```
$ORACLE_HOME/oracle_common/common/bin/unpack.sh \
-domain=user_projects/domains/base_domain2 \
-template=/tmp/base_domain_template.jar \
-app_dir=user_projects/applications/base_domain2
```

6.10 Starting the Node Manager

To start Node Manager, run the following:

```
$DOMAIN_HOME/bin/startNodeManager.sh &
```

When you use machine scoped Node Manager, see "Using Node Manager" in *Administering Node Manager for Oracle WebLogic Server* for more information on Node Manager start options.

6.11 Starting the Managed Servers

To use the Administration Console to start the Managed Servers:

1. In the left pane of the Console, expand **Environment** and select **Servers**.
2. In the **Servers** table, click the name of the Managed Server that you moved to the new machine to start it.
3. Select **Control > Start/Stop**.
4. In the **Server Status** table, select the check box next to the name of the server(s) you want to start and click **Start**.
5. On the Server Life Cycle Assistant page, click **Yes** to confirm.

See Also: To use WLST commands or Fusion Middleware Control to start Managed Servers, see "Starting and Stopping Managed Servers" in the guide *Oracle Fusion Middleware Administering Oracle Fusion Middleware*.

6.12 Verifying Machine Scale Out

To determine if the machine scale out succeeded, verify that the server status has changed to **RUNNING** after you start it using the Administration Console.

Part III

Component Procedures

Part III describes procedures that are unique to certain component products.

This part includes the following chapters:

- Chapter 7, "Configuring High Availability for Web Tier Components"
- Chapter 8, "Configuring High Availability for Oracle Application Development Framework"
- Chapter 9, "Configuring High Availability for Other Components"

7

Configuring High Availability for Web Tier Components

The Web Tier of Oracle Fusion Middleware is the outermost tier in the architecture, closest to the end user. The key component of the Web Tier is Oracle HTTP Server. This chapter describes high availability concepts and configuration procedures for Oracle HTTP Server and includes the following topics:

- Section 7.1, "Oracle HTTP Server and High Availability Concepts"
- Section 7.2, "Oracle HTTP Server Single-Instance Characteristics"
- Section 7.3, "Oracle HTTP Server Startup and Shutdown Lifecycle"
- Section 7.4, "Starting and Stopping Oracle HTTP Server"
- Section 7.5, "Oracle HTTP Server High Availability Architecture and Failover Considerations"
- Section 7.6, "Oracle HTTP Server Protection from Failures and Expected Behaviors"
- Section 7.7, "Oracle HTTP Server Cluster-Wide Configuration Changes"
- Section 7.8, "Configuring Oracle HTTP Server for High Availability"

7.1 Oracle HTTP Server and High Availability Concepts

Oracle HTTP Server (OHS) is the Web server component for Oracle Fusion Middleware. It provides a listener for Oracle WebLogic Server and the framework for hosting static pages, dynamic pages, and applications over the Web.

See Also: For more information on working with OHS, see the following:

- "Managing Oracle HTTP Server" in the guide *Administering Oracle HTTP Server*. This section includes topics such as "Performing Basic OHS Tasks," "Creating an OHS Instance," and "Managing and Monitoring Server Processes.
- *Oracle Fusion Middleware Installing and Configuring Oracle HTTP Server*

7.2 Oracle HTTP Server Single-Instance Characteristics

Oracle HTTP Server is based on Apache infrastructure and includes modules developed by Oracle that you can use to extend OHS's core functionality. OHS has the following components to handle client requests:

- **HTTP listener**, to handle incoming requests and route them to the appropriate processing utility.
- **Modules (mods)**, to implement and extend the basic functionality of Oracle HTTP Server. Many of the standard Apache modules are included with Oracle HTTP Server. Oracle also includes several modules that are specific to Oracle HTTP Server to support integration between Oracle HTTP Server and other Oracle HTTP Server components.

Oracle HTTP Server can also be a proxy server, both forward and reverse. A reverse proxy enables content served by different servers to appear as if it comes from one server.

7.2.1 Oracle HTTP Server and Oracle WebLogic Server

Oracle HTTP Server does not require an Oracle WebLogic domain but is usually used in conjunction with one. Oracle recommends associating Oracle HTTP Server with the WebLogic domain because it enables Oracle HTTP Server to be incorporated into the Administration Console for centralized management and monitoring.

The link to WebLogic Managed Servers is handled through the `mod_wl_ohs` module. This module is configured by routing requests of a particular type, for example, JSPs, or by routing requests destined to a URL to specific Managed Servers.

Typically you use Oracle HTTP Server to front end a cluster of WebLogic servers. When Oracle HTTP Server front-ends a cluster of WebLogic servers, a special `mod_wl_ohs` directive, WebLogicCluster, specifies a comma-separated list of cluster members. The process works as follows:

1. When `mod_wl_ohs` receives a request requiring a Managed Server, it sends that request to one of the WebLogic cluster members listed in the directive. At least one server must be available to service the request.
2. When the Managed Server receives the request, it processes it and sends a complete list of cluster members back to `mod_wl_ohs`.
3. When `mod_wl_ohs` receives the updated list, it dynamically adds any previously unknown servers to the list of known servers, enabling all future requests to be load balanced across the full cluster member list. This process has the advantage of enabling new Managed Servers to be added to the cluster without updating `mod_wl_ohs` or adding the Oracle HTTP Server

> **Note:** You do not need to include all the current Managed Servers in the `mod_wl_ohs` directive when you start; a high availability setup requires only two cluster members in the list for the first call to work. See Configuring the WebLogic Proxy Plug-In for Oracle HTTP Server in the guide *Oracle Fusion Middleware Using Oracle WebLogic Server Proxy Plug-Ins* for more information on running a high availability deployment of Oracle HTTP Server.

See Also: For more information on Oracle WebLogic clusters, see *Oracle Fusion Middleware Using Clusters for Oracle WebLogic Server*.

7.3 Oracle HTTP Server Startup and Shutdown Lifecycle

After Oracle HTTP Server starts, it is ready to listen for and respond to HTTP(S) requests.

The request processing model on Microsoft Windows systems differs from that on UNIX systems:

- For Microsoft Windows, there is a single parent process and a single child process. The child process creates threads that are responsible for handling client requests. The number of created threads is static and can be configured for performance.
- For UNIX, there is a single parent process that manages multiple child processes. The child processes are responsible for handling requests. The parent process brings up additional child processes as necessary, based on configuration.

See Also: For more information on the OHS processing model, see "Oracle HTTP Server Processing Model" in the *Administrator's Guide for Oracle HTTP Server*.

7.4 Starting and Stopping Oracle HTTP Server

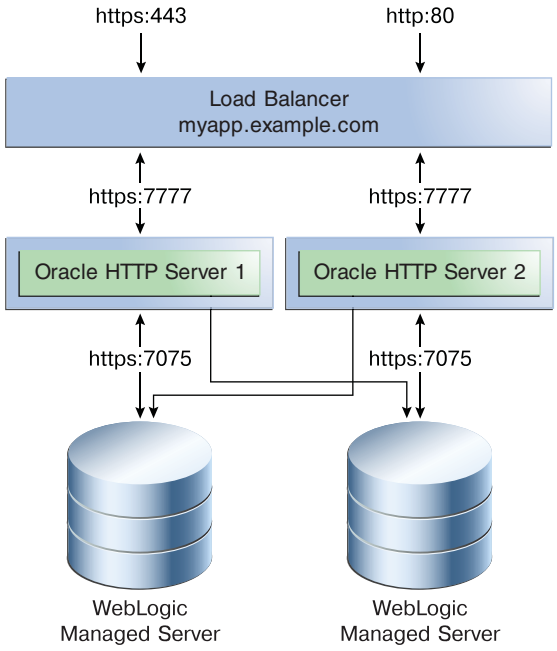
You can use Fusion Middleware Control or the WebLogic Scripting Tool (WLST) to start, stop, and restart Oracle HTTP Server. If you plan to use WLST, you should familiarize yourself with that tool; see "Getting Started Using the Oracle WebLogic Scripting Tool (WLST)" in the *Oracle Fusion Middleware Administrator's Guide*.

See Also: For information on starting and stopping OHS, see Starting, Stopping, and Restarting Oracle HTTP Server in the *Administrator's Guide for Oracle HTTP Server*.

7.5 Oracle HTTP Server High Availability Architecture and Failover Considerations

Figure 7–1 shows two Oracle HTTP Servers placed behind a load balancer.

Figure 7–1 Oracle HTTP Server High Availability Architecture



The load balancer receives requests from users and forwards them on to the connected Oracle HTTP Servers. In Figure 7–1, the load balancer receives the requests on the standard HTTP/HTTPS ports (80/443). However, it then passes the requests on to the Oracle HTTP Servers using completely different ports. This setup has the following advantages:

- Actual ports are hidden from users.
- Users do not have to add the port numbers to the URL.

On UNIX-based systems, it is not mandatory to start Oracle HTTP Server with root privileges. Only root can start a process which uses a port less than 1024.

The load balancer routes requests to the functioning Oracle HTTP Servers.

Figure 7–1 also shows how Oracle HTTP Server distributes requests to WebLogic Managed Servers. For high availability, it is assumed that each pair of components (Oracle HTTP Server and Web Logic Managed Servers) exist on different host computers.

You can separate this architecture across two servers. Alternatively, in more complex implementations, each component can reside on a completely separate server.

7.6 Oracle HTTP Server Protection from Failures and Expected Behaviors

There are two categories of Oracle HTTP Server failures: **process failures** and **node failures**. An individual operating system process may fail. A node failure could involve failure of the entire host computer that Oracle HTTP Server runs on. This section describes failure types and the systems or processes that take over if they occur.

Process Failure

Node manager protects and manages Oracle HTTP Server processes. If an Oracle HTTP Server process fails, Node Manager automatically restarts the process.

Node Failure

If an entire node fails, the load balancer in front of Oracle HTTP Server sends a request to another Oracle HTTP Server if the first one does not respond, or is determined to be failed through URL pings.

WebLogic Managed Server Failure

In a high availability deployment, Oracle WebLogic Managed Servers are part of a cluster. If one of the Managed Servers fails, `mod_wl_ohs` automatically redirects requests to one of the active cluster members. If the application stores state, state replication is enabled within the cluster, which enables redirected requests access to the same state information.

Database Failure

Typically, database failures are an issue only when `mod_oradav` or `mod_plsql` is used. If this is an Oracle Real Application Clusters (Oracle RAC) database, the failure characteristics are determined by the defined Oracle RAC connection.

If client connection failover is configured, any in-flight transactions are rolled back, and a database reconnection is required.

If Transparent Application Failover (TAF) is configured, any in-flight database write is rolled back but an automatic database reconnection takes place and select statements are automatically recovered. In this scenario, TAF fails over select statements only; package variables are lost. (TAF is a feature of the Java Database Connectivity (JDBC) Oracle Call Interface driver. It enables the application to automatically reconnect to a database, if the database instance to which the connection is made fails. In this case, the active transactions roll back.

7.7 Oracle HTTP Server Cluster-Wide Configuration Changes

Oracle HTTP Servers are managed independently by Fusion Middleware Control. The Oracle HTTP server configuration is file-based; if you make changes to one Oracle HTTP Server, you must manually copy the changes to other Oracle HTTP Servers in the configuration. This also applies to static HTML files stored in the htdocs directory.

7.8 Configuring Oracle HTTP Server for High Availability

This section describes how to configure an example high availability deployment of Oracle HTTP Server.

This section includes the following topics:

- Section 7.8.1, "Prerequisites"
- Section 7.8.2, "Installing Oracle HTTP Server on WEBHOST2"
- Section 7.8.3, "Configuring and Validating the OHS High Availability Deployment"

7.8.1 Prerequisites

Review the following prerequisites before configuring a high availability Oracle HTTP Server deployment.

- Section 7.8.1.1, "Configuring the Load Balancer"
- Section 7.8.1.2, "Installing Oracle HTTP Server on WEBHOST1"

- Section 7.8.1.3, "Configuring Virtual Host(s)"
- Section 7.8.1.4, "Configuring mod_wl_ohs.conf"

7.8.1.1 Configuring the Load Balancer

To distribute requests against Oracle HTTP Servers, you must use an external load balancer to distribute HTTP(S) requests between available Oracle HTTP Servers. If you have an external load balancer, it must have the features that Section 2.1.1, "Third-Party Load Balancer Requirements" describes, including:

- Virtual server name and port configuration
- Process failure detection
- Monitoring of ports (HTTP, HTTPS) for Oracle HTTP and HTTPS
- SSL Protocol Conversion (if required)

The following sections describe steps to configure your load balancer:

- "Configuring Virtual Server Names and Ports for the Load Balancer"
- "Managing Port Numbers"

Configuring Virtual Server Names and Ports for the Load Balancer

For each application, such as `myapp.example.com`, configure the load balancer with a virtual server name and associated ports. In an Oracle HTTP Server installation, these virtual servers are configured for HTTP connections, which are distributed across the HTTP servers.

If your site is serving requests for HTTP and HTTPS connections, Oracle recommends that HTTPS requests terminate at the load balancer and pass through as HTTP requests. To do this, the load balancer should be able to perform the protocol conversion and must be configured for persistent HTTP sessions.

This example configuration assumes that the load balancer is configured as:

- **Virtual Host:** `Myapp.example.com`
- **Virtual Port:** `7777`
- **Server Pool:** `Map`
- **Server:** `WEBHOST1, Port 7777, WEBHOST2, Port 7777`

Managing Port Numbers

Many Oracle Fusion Middleware components and services use ports. As an administrator, you must know the port numbers that the services use and ensure that two services are not configured to use the same port number on your host computer.

Most port numbers are assigned during installation. It is important that any traffic going from the Oracle HTTP Servers to the Oracle WebLogic Servers has access through any firewalls.

7.8.1.2 Installing Oracle HTTP Server on WEBHOST1

To install Oracle HTTP Server on WEBHOST1, see the steps in "Installing the Oracle HTTP Server Software" in the guide *Oracle Fusion Middleware Installing and Configuring Oracle HTTP Server*.

Validating the OHS Installation

Validate the installation using the following URL to access the Oracle HTTP Server home page:

```
http://webhost1:7777/
```

7.8.1.3 Configuring Virtual Host(s)

For each virtual host or site name that you use, add an entry to the Oracle HTTP Server configuration. Create a file named `virtual_hosts.conf` located in the *`DOMAIN_HOME`*`/config/fmwconfig/components/OHS/`*`ohs_component_name`*`/moduleconf` directory as follows:

```
NameVirtualHost *:7777
<VirtualHost *:7777>
    ServerName http://myapp.example.com:80
    RewriteEngine On
    RewriteOptions inherit
    UseCanonicalName On
</VirtualHost>
```

If you are using SSL/SSL Termination (*):

```
NameVirtualHost *:7777
<VirtualHost *:7777>
    ServerName https://myapp.example.com:443
    RewriteEngine On
    RewriteOptions inherit
    UseCanonicalName On
</VirtualHost>
```

7.8.1.4 Configuring mod_wl_ohs.conf

After you install and configure Oracle HTTP Server, link it to any defined WebLogic Managed Servers by editing the `mod_wl_ohs.conf` file located in *`DOMAIN_HOME`*`/config/fmwconfig/components/OHS/instances/`*`componentName`* directory.

See "Configuring the WebLogic Proxy Plug-In for Oracle HTTP Server" in the guide *Oracle Fusion Middleware Using Oracle WebLogic Server Proxy Plug-Ins 12.1.2* for more information on editing the `mod_wl_ohs.conf` file.

The following is an example of `mod_wl_ohs.conf` entries:

```
LoadModule weblogic_module PRODUCT_HOME/modules/mod_wl_ohs.so

<IfModule mod_weblogic.c>
      WebLogicCluster apphost1.example.com:7050, apphost2.example.com:7050
      MatchExpression *.jsp
 </IfModule>

<Location /weblogic>
  SetHandler weblogic-handler
  WebLogicCluster apphost1.example.com:7050,apphost2.example.com:7050
  DefaultFileName index.jsp
</Location>
```

Note: If you use SSL termination AND route requests to WebLogic, then the following additional configuration is required.

In the WebLogic console, WebLogic Plugin Enabled must be set to true, either at the domain, cluster or Managed Server level.

In the Location block which directs requests to the WebLogic Managed Servers, the following lines also need to be added.

```
WLProxySSL ON
WLProxySSLPassThrough ON
```

For example:

```
<Location /weblogic>
  SetHandler weblogic-handler
  WebLogicCluster
apphost1.example.com:7050,apphost2.example.com:7050
  WLProxySSL On
  WLProxySSLPassThrough ON
  DefaultFileName index.jsp
</Location>
```

After you enable the WebLogic plugin, restart the Administration Server.

These examples show two different ways of routing requests to Oracle WebLogic Managed Servers:

- The `<ifModule>` block sends any requests ending in *.jsp to the WebLogic Managed Server cluster located on Apphost1 and Apphost2.
- The `<Location>` block sends any requests with URLs prefixed by /weblogic to the WebLogic Managed Server cluster located on Apphost1 and Apphost2.

7.8.2 Installing Oracle HTTP Server on WEBHOST2

To install Oracle HTTP Server on WEBHOST2, see the steps in the topic "Installing the Oracle HTTP Server Software" in the guide *Oracle Fusion Middleware Installing and Configuring Oracle HTTP Server*.

Validating the OHS Installation

Validate the installation on WEBHOST2 by using the following URL to access the Oracle HTTP Server home page:

```
http://webhost2:7777/
```

7.8.3 Configuring and Validating the OHS High Availability Deployment

To configure and validate the OHS high availability deployment, follow these steps:

- Section 7.8.3.1, "Configuring Virtual Host(s)"
- Section 7.8.3.2, "Validating the Oracle HTTP Server Configuration"

7.8.3.1 Configuring Virtual Host(s)

Add an entry for each virtual host or site name to the Oracle HTTP Server configuration. Copy the file `virtual_hosts.conf` from WEBHOST1 to WEBHOST2.

The file is located in the directory DOMAIN_HOME/config/fmwconfig/components/OHS/ohs_name/moduleconf

7.8.3.2 Validating the Oracle HTTP Server Configuration

Validate the configuration by using the following URLs:

http://myapp.example.com/

https://myapp.example.com (if using SSL/SSL termination)

http://myapp.example.com:7777/weblogic

8

Configuring High Availability for Oracle Application Development Framework

This chapter describes procedures specific to Oracle Application Development Framework (ADF).

> **Note:** Unlike the other chapters in this guide, this chapter is written for ADF developers rather than administrators.

This chapter includes the following topics:

- Section 8.1, "Oracle ADF High Availability Considerations"
- Section 8.2, "Configuring Oracle ADF for High Availability"
- Section 8.3, "Troubleshooting Oracle ADF High Availability"

> **See Also:** For more information on ADF, see the following:
>
> - Oracle ADF Key Concepts in *Understanding the Oracle Application Development Framework*
> - *Oracle Fusion Middleware Administering Oracle ADF Applications*

8.1 Oracle ADF High Availability Considerations

Fusion web applications built on the Oracle ADF technology stack are Java EE applications (and J2EE applications). This section includes the following topics:

- Section 8.1.1, "Oracle ADF Scope and Session State"
- Section 8.1.2, "Oracle ADF Failover and Expected Behavior"
- Section 8.1.3, "Configuring the ADF Application Module for Oracle RAC"

8.1.1 Oracle ADF Scope and Session State

At runtime, ADF objects such as the binding container and managed beans are instantiated. Each of these objects has a defined life span set by its scope attribute.

For more information, see "About Object Scope Lifecycles" in the guide *Oracle Fusion Middleware Developing Fusion Web Applications with Oracle Application Development Framework* for more information.

8.1.2 Oracle ADF Failover and Expected Behavior

An Oracle WebLogic cluster provides application high availability. If one member of the cluster is unavailable, any other available member of the cluster is able to handle the request. This topic describes failover requirements and the expected behavior of an application in the event of a failover.

Session Failover Requirements

For seamless failover of a Fusion web application, the application must meet the following conditions:

- The application is in a cluster and at least one member of the application cluster is available to serve the request.
- For stateful applications, state replication is configured correctly as Section 8.2, "Configuring Oracle ADF for High Availability" describes.
- If you use Oracle HTTP Server, the server is configured with the WebLogicCluster directive to balance among all available application instances.
- If you use a hardware load balancer, the load balancer is:
  - Routing traffic to all available instances
  - Configured correctly with a health monitor to mark unavailable instances
  - Configured to support persistence of session state

Expected Behavior for Application Failover

If the environment is configured correctly, application users do not notice when an application instance in a cluster becomes unavailable. The sequence of events in an application failover is, for example, as follows:

1. A user makes a request and is routed by a hardware load balancer to Instance A of the application.
2. Instance A of the application becomes unavailable because of node failure, process failure, or network failure.
3. The hardware load balancer marks Instance A as unavailable.
4. The user makes a subsequent request. The request is routed to Instance B.
5. Instance B is configured as a replication partner of Instance A and has the user's session state.
6. The application resumes using the session state on Instance B and the user continues working without interruption.

Oracle JRF Asynchronous Web Services (Pinned Service Behavior)

When you use Oracle JRF Asynchronous Web Services, the asynchronous web service is pinned to a service and does not fail over. When you use a reliability protocol such as WS-RM, the higher-level protocol reconnects to a new server after a failure.

For more information on Oracle JRF Asynchronous Web Services, see the *Domain Template Reference*.

8.1.3 Configuring the ADF Application Module for Oracle RAC

When you configure the ADF application module to access a highly available database system, such as redundant databases or Oracle Real Application Clusters (Oracle RAC) as the backend, the data source must be container-defined. In this scenario, you

must use a GridLink data source or a multi data source. However, from the standpoint of the application module configuration, the naming convention for the multi data source or GridLink data source is the same as it is for a non-multi data source or GridLink data source. This naming convention ensures that the correct data source is used at runtime.

To configure GridLink data sources for high availability applications, see Section 4.4, "Configuring Active GridLink Data Sources with Oracle RAC." For multi data sources, see Section 4.5.1, "Configuring Multi Data Sources with Oracle RAC."

Modifying the mdsDS Data Source URL

When you use a GridLink data source or a multi data source with a customer-provided ADF application, you must change the mdsDS data source URL in the Weblogic Server Administration Console after you start the Administration Server.

To modify the mdsDS data source URL:

1. Log into the WebLogic Server Administration Console. Select **Services** then **Data Sources**.
2. Select **mdsDS** then **Connection Pool**.
3. Change the JDBC URL to specify it in the format:

   ```
   jdbc:oracle:thin:@hostname:port/service-name
   ```

   For example, if the JDBC URL is:

   ```
   jdbc:oracle:thin:@(DESCRIPTION=(ENABLE=BROKEN)(ADDRESS_
   LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=slc00erterw-r.us.example.com)(PORT=1521)))(CO
   NNECT_DATA=(SERVICE_NAME=srv2_koala.us.example.com)))
   ```

   Replace the URL with the following:

   ```
   jdbc:oracle:thin:@slc00erterw-r.us.example.com:1521/srv2_koala.us.example.com
   ```

4. Save the changes then restart all servers, including the Administration Server.

8.2 Configuring Oracle ADF for High Availability

To support automatic replication and failover for web applications in a clustered environment, Oracle WebLogic Server supports mechanisms for replicating HTTP session state across clusters. You can configure Oracle ADF to ensure the Fusion web application's state can be restored from any server in the cluster.

This section includes the following topics:

- Section 8.2.1, "Configuring Application Modules"
- Section 8.2.2, "Configuring weblogic.xml"
- Section 8.2.3, "Configuring adf-config.xml"
- Section 8.2.4, "Configuring org.apache.myfaces.trinidad.CHECK_FILE_MODIFICATION"

8.2.1 Configuring Application Modules

An **application module** is the transactional component that UI clients use to work with application data. It defines an updateable data model and top-level procedures and functions, called *service methods*, related to a logical unit of work related to an

end-user task. An application module supports *passivating*, or storing, its transaction state as a snapshot in the database. It also supports the reverse operation of activating the transaction state from one of these saved snapshots.

For more information on application module state management, see "Introduction to Fusion Web Application State Management" in *Developing Fusion Web Applications with Oracle Application Development Framework*.

To enable support for ADF Business Components failover, set the `jbo.dofailover` parameter to `true` so that the application module state is saved on release. This enables Oracle ADF to restore the application module state from a snapshot saved from a previous check in. By contrast, when the failover feature is disabled, which it is by default, then application module state is saved only when the application is reused by a subsequent user session and only when the application module pool cannot find an unused application module.

You can set this parameter in your application module configuration on the **Pooling and Scalability** tab of the Edit Business Components Configuration dialog.

To configure application modules for high availability:

1. Launch JDeveloper and open the application.
2. In the Application Navigator, expand the project that contains the data model and locate the application module.
3. Right-click the application module and select **Configurations**.
4. Click **Edit**.
5. Click the **Pooling and Scalability** tab.
6. Select the **Failover Transaction State Upon Managed Release** checkbox.
7. Click **OK** to close the **Edit Business Components Configuration** dialog.
8. Click **OK** to close the **Manage Configurations** dialog.

8.2.2 Configuring weblogic.xml

To enable support for replicating HTTP session state, you must assign a value to the `persistent-store-type` element in the Oracle WebLogic Server `weblogic.xml` file. The value `replicated_if_clustered` ensures that the in-effect persistent store type will be replicated so that sessions on the clustered environment are stored in accordance with the value set for the cluster of servers to which this server belongs.

> **Note:** Oracle ADF applications such as the Oracle WebCenter Portal Suite are preconfigured and do not need additional configuration.

To configure the `weblogic.xml` file for high availability:

1. Launch JDeveloper and open the application.
2. In the Application Navigator, expand the project that contains the web application and expand the **WEB-INF** folder.
3. Double-click the **weblogic.xml** file, and click the **Source** tab to edit the file.
4. In the file, add the `persistent-store-type` definition to the `session-descriptor` element:

```
<weblogic-web-app>
    <session-descriptor>
```

```
    <persistent-store-type>
    replicated_if_clustered
    </persistent-store-type>
    </session-descriptor>
</weblogic-web-app>
```

8.2.3 Configuring adf-config.xml

When you design an application to run in a clustered environment, you must ensure that Oracle ADF is aware of changes to managed beans stored in ADF scopes (view scope and page flow scope).

When a value within a managed bean in either view scope or page flow scope is modified, the application must notify Oracle ADF so that it can ensure the bean's new value is replicated.

To enable ADF Controller to track changes to ADF memory scopes and replicate the page flow scope and view scope within the server cluster, you must set the ADF Controller parameter `<adf-scope-ha-support>` in the application's `adf-config.xml` file to `true`. For example, when set to `true` for an application and that application adds or removes a bean from a page flow scope during a request, the change will automatically replicated within a cluster.

The `adf-config.xml` file is the central configuration file for all ADF components. The file contains sections to configure the runtime behavior for ADF components, including, ADF Controller.

Note: If your application uses MDS and will use an Oracle database that supports failover, Oracle recommends enabling MDS retry on failover. To do this, add the following `retry-connection` entry to the MDS configuration section of adf-config.xml.

```
<persistence-config>
         <metadata-namespaces>...
         <metadata-store-usages>...
         <external-change-detection enabled="false" />
         <read-only-mode enabled="true"/>
         <retry-connection enabled="true"/>
</persistence-config>
```

To configure the `adf-config.xml` file for high availability:

1. Launch JDeveloper and open the application.
2. In the Application Navigator, expand the **Application Resources**.
3. Select **Descriptors**, and then select the **ADF META-INF** node.
4. Double-click the **adf-config.xml** file, and click the **Source** tab to edit the file.
5. Add the following to the file:

```
<adf-controller-config xmlns="http://xmlns.example.com/adf/controller/config">
 <adf-scope-ha-support>true</adf-scope-ha-support>
</adf-controller-config>
```

For more information about using the `adf-config.xml` file to configure ADF, see the chapter on creating complex task flows in the guide *Developing Fusion Web Applications with Oracle Application Development Framework*.

8.2.4 Configuring org.apache.myfaces.trinidad.CHECK_FILE_MODIFICATION

The `org.apache.myfaces.trinidad.CHECK_FILE_MODIFICATION` parameter must not be set to `true` when it runs in a high availability environment. Setting this context parameter to true can lead to errors after failover occurs. Note that this limitation should not affect your work because you typically use this parameter in a development environment only, not a high availability environment.

8.3 Troubleshooting Oracle ADF High Availability

This section describes procedures for troubleshooting possible issues with Oracle ADF. Topics in this section include the following:

- Section 8.3.1, "Troubleshooting Oracle ADF Development Issues"
- Section 8.3.2, "Troubleshooting Oracle ADF Deployment Issues"
- Section 8.3.3, "Troubleshooting Oracle ADF Replication and Failover Issues"

8.3.1 Troubleshooting Oracle ADF Development Issues

When you develop the Fusion web application in Oracle JDeveloper, the integrated development environment provides support for detecting potential High Availability issues. The warnings that JDeveloper provides are generated by the audit framework and are triggered to appear in the JDeveloper source editors. The warnings the editors show are based on the audit rules for High Availability applications.

The High Availability audit rules that JDeveloper enables by default are:

- **ADF Controller Configuration - High Availability for ADF Scopes is not Enabled** warns the developer that the `adf-scope-ha-support` flag in the `adf-config.xml` file is set is not set to `true`. This audit rule fires only when the `<adf-controller-config>` element is in the ADF application-level configuration file (`adf-config.xml`).
- **ADF Page Flows - Bean in Scope Map is Modified** warns the developer when the some code calls a setter method on a bean to indicate that the code did not subsequently call the `ControllerContext.markScopeDirty()` method. This audit rule fire only when the `adf-scope-ha-support` flag in the `adf-config.xml` file is set to true.
- **ADF Page Flows - EL Bean is Modified** warns the developer when some code evaluates an EL expression that mutates a bean to indicate that the code did not subsequently call the `ControllerContext.markScopeDirty()` method. This audit rule fire only when the `adf-scope-ha-support` flag in the `adf-config.xml` file is set to true.
- **ADF Page Flows - Managed Bean Class Not Serializable** warns the developer that a managed bean has a non-serializable class defined in `viewScope`, `pageFlowScope`, or `sessionScope`. This audit rule fire only when the `adf-scope-ha-support` flag in the `adf-config.xml` file is set to `true`.

You can modify the High Availability audit rule settings using the Preference dialog in JDeveloper. From the JDeveloper toolbar, choose **Tools - Preferences**, under **Audit - Profiles** expand **ADF Controller Configuration** or **ADF Pages Flows** and make the desired audit rule selections.

You can also trigger the audit by choosing **Build - Audit** ***project.jpr*** from the JDeveloper toolbar.

8.3.2 Troubleshooting Oracle ADF Deployment Issues

There are two actions to take if you encounter ADF deployment issues: verify the JRF Runtime and verify the status of application deployments.

8.3.2.1 Verifying the JRF Runtime Installation

The first step to troubleshooting an ADF deployment is to verify that the JRF Runtime is installed on the WebLogic Server domain. ADF applications require the JRF and ADF runtime. You cannot run ADF applications with a standalone WebLogic Server domain or just the WebLogic Server portion of the Application Development product; you must extend it with the JRF extension template.

For more information on the JRF Template, see "Oracle JRF and ADF Templates" in the *Oracle Fusion Middleware Domain Template Reference*.

8.3.2.2 Verifying the Success of All Application Deployments

Fusion web applications deploy when the Managed Server first starts. Use the Administration Console to check that all application deployments were successful:

1. Click **Deployments** in the left hand pane. The right hand pane shows the application deployments and their status. The state of all applications, assuming all the servers are running, should be ACTIVE.
2. If an application deployment has failed, the server logs may provide some indication of why the application was not deployed successfully. The server logs are located in the *DOMAIN_HOME*/servers/*server_name*/logs directory. Common issues include:

- Unavailability of external resources, such as database resources. Examine the error, fix it, and attempt to redeploy the application.
- The appropriate applications or libraries are not targeted correctly to the right Managed Server or Cluster.

8.3.3 Troubleshooting Oracle ADF Replication and Failover Issues

State Replication is most prominent in failover scenarios. A user working on one server may discover that, upon failover:

- Windows may close or the state might reset.
- Screens may require a reset.
- The application may redirect to the logon screen.

To diagnose and troubleshoot state replication issues.

1. Confirm that this is not a known replication issue.

   See Section 8.1.2, "Oracle ADF Failover and Expected Behavior" for possible expected behaviors. Before proceeding to further diagnose the issue, first confirm that the failover behavior is not an expected behavior.

2. Check load balancer settings.

   For replication and failover to function correctly, the load balancer must be configured with the appropriate persistence settings. For more details on configuring Hardware Load Balancers for Oracle WebLogic Server, see *Oracle Fusion Middleware Using Clusters for Oracle WebLogic Server*.

3. Check the cluster status.

Replication occurs within the context of a cluster. For failover to be successful, there must be at least one other healthy member of the cluster available. You can check cluster status in one of two ways:

- Check the cluster status using the Oracle WebLogic Server Administration Console - In the Left-hand pane, click on **Servers**. Verify the state of all servers in the cluster.
- Check the cluster status using the `weblogic.Admin` utility. You can use the `weblogic.Admin` utility to query the state of all servers in a specific cluster. For example:

```
$ java weblogic.Admin -url Adminhost:7001 -username <username> -password
 <password>  CLUSTERSTATE -clustername Spaces_Cluster
```

This example returns:

```
There are 2 server(s) in cluster: Spaces_Cluster
The alive servers and their respective states are listed below:
Application Server---RUNNING
Managed Server---RUNNING
```

4. Check cluster communications.

Even if all cluster members are running, there may be communication issues which prevent them from communicating replication information to each other. There are two types of cluster communication configurations. Troubleshooting depends on the cluster type:

- Checking Unicast cluster communications - For Unicast clusters, Managed Servers must be able to access each other's hosts and each other's default listening port.

  Ensure that all individual Managed Servers have their Listen Address set correctly. You can find this setting by selecting **Configuration**, **General** for each Managed Server.
- Checking Multicast cluster communications - For multicast clusters, servers must be able to intercept the same multicast traffic. Ensure that multicast is configured correctly by running the WebLogic utility `utils.MulticastTest` on each machine. For example:

```
$ java utils.MulticastTest -H
```

5. Confirm Oracle WebLogic Server application configuration.

Oracle WebLogic Server is not configured by default for failover. In-memory replication takes place only with the proper setting in the `weblogic.xml` file:

```
<session-descriptor>
<persistent-store-type>replicated_if_clustered</persistent-store-type>
</session-descriptor>
```

A persistent-store-type of replicated is also acceptable. This setting can be made in JDeveloper, as described in Section 8.2.2, "Configuring weblogic.xml."

6. Confirm Oracle ADF Business Components configuration.

Oracle ADF is not configured by default for failover. Failover is supported only with the proper setting in the ADF Business Components configuration file (`bc4j.xcfg`):

```
<AppModuleConfig ...
 <AM-Pooling jbo.dofailover="true"/>
```

```
</AppModuleConfig>
```

This setting is made in JDeveloper through the Edit Business Components Configuration dialog, as described in Section 8.2.1, "Configuring Application Modules."

7. Confirm Oracle WebLogic Server connection pool parameter.

   Set an appropriate value for the `weblogic-application.xml` deployment descriptor parameter `inactive-connection-timeout-seconds` on the element `<connection-check-params>` `pool-params`.

   When enabling application module state passivation, a failure can occur when Oracle WebLogic Server is configured to forcibly release connections back into the pool. The failure creates an exception "`Connection has already been closed`" that gets saved to the server log. The user interface does not show this exception.

   Set `inactive-connection-timeout-seconds` to several minutes. In most cases, this setting avoids forcing the inactive connection timeout and passivation failure. Adjust the setting as needed for your environment.

8. Confirm ADF Controller configuration.

   Oracle ADF is not configured by default to replicate changes to ADF objects in ADF memory scopes. ADF object replication is supported only with the proper setting in the ADF application-level configuration file (`adf-config.xml`):

   ```
   <adfc:adf-controller-config>
    <adfc:adf-scope-ha-support>true</adfc:adf-scope-ha-support>
   </adfc:adf-controller-config>
   ```

   This setting is made in JDeveloper through the source editor. See Section 8.2.1, "Configuring Application Modules."

9. Check default logger messages.

   By default the ADF log shows high-level messages (`INFO` level). The default logging often reports problems with serialization and replication without the need to enable more detailed log messages.

10. Enable log messages for ADF high availability applications.

    Configure the ADF logger to output runtime messages for high availability. By default the ADF log shows high-level messages (`INFO` level). You enable high availability diagnostics for ADF Controller by setting the logging level in Fusion Middleware Control to `FINE`.

    When enabled, the logger outputs a warning if the `adfc:adf-scope-ha-support` setting in the `adf-config.xml` file is not set.

11. Enable debug.

    Check the server logs for any unusual messages on Managed Server startup. In particular, if the Managed Server is unable to locate other members of the cluster. The server logs are located in the *`DOMAIN_HOME`*`/servers/`*`SERVER_NAME`*`/logs` directory.

    For further debugging, enable the flags `DebugCluster`, `DebugClusterAnnouncements`, `DebugFailOver`, `DebugReplication`, and `DebugReplicationDetails`. You can enable each flag with the `weblogic.Admin` utility:

    ```
    $ java weblogic.Admin -url Adminhost:7001 -username <username> -password
     <password> SET -type ServerDebug -property DebugCluster true
    ```

12. Enable component state serialization checking.

    Enable server checking to ensure no unserializable state content on session attributes is detected. This check is disabled by default to reduce runtime overhead. Serialization checking is supported by the Java server system property `org.apache.myfaces.trinidad.CHECK_STATE_SERIALIZATION`.

    Table 8–1 shows the options you can use with the property. Use commas to delimit the options.

Table 8–1 CHECK_STATE_SERIALIZATION Options

Option	Description
tree	Checks whether the entire component tree is serializable. This is the fastest component check. Most testing should be performed with this flag enabled.
component	Checks each component individually for serializability. This option is much slower than "tree." It is typically turned on only after testing with "tree" reports an error. This option narrows down the problematic component.
property	Checks each component attribute individually for serializability. This is slower than "component" and narrows down the specific problematic component attribute after a failure has been detected in the "tree" or "component" modes.
session	Checks that all attributes in the JSF Session Map that are marked as Serializable are serializable.
application	Checks that all attributes in the JSF Application Map that are marked as Serializable are serializable.
beans	Checks that any serializable object in the appropriate map has been marked as dirty if the serializable content of the object changes during the request.
all	Checks everything.

    For high availability testing, start off by validating that the Session and JSF state is serializable by launching the application server with the system property:

    ```
    -Dorg.apache.myfaces.trinidad.CHECK_STATE_SERIALIZATION=session,tree
    ```

    Add the `beans` option to check that any serializable object in the appropriate map has been marked as dirty if the serialized content of the object has changed during the request:

    ```
    -Dorg.apache.myfaces.trinidad.CHECK_STATE_SERIALIZATION=session,tree,beans
    ```

    If a JSF state serialization failure is detected, relaunch the application server with the system property to enable component and property flags and rerun the test:

    ```
    -Dorg.apache.myfaces.trinidad.CHECK_STATE_SERIALIZATION=all
    ```

    These are Java system properties and you must specify them when you start the application server.

9

Configuring High Availability for Other Components

This chapter describes information unique to certain component products.

For this release, this chapter includes the following topics:

- Section 9.1, "Deploying the Oracle Virtual Assembly Builder Deployer."
- Section 9.2, "Deploying Oracle Data Integrator"

9.1 Deploying the Oracle Virtual Assembly Builder Deployer

You can deploy the Oracle Virtual Assembly Builder Deployer to a single server only, not a cluster. You can deploy it to an Administration Server in a single server domain.

> **See Also:** For more information on Oracle Virtual Assembly Builder, see *Using Oracle Virtual Assembly Builder*.

9.2 Deploying Oracle Data Integrator

This topic describes considerations for configuring Oracle Data Integrator repository connections to Oracle Real Application Clusters:

- Section 9.2.1, "Oracle RAC Retry Connectivity for Source and Target Connections"
- Section 9.2.2, "Configuring Repository Connections to Oracle RAC"

9.2.1 Oracle RAC Retry Connectivity for Source and Target Connections

When you configure Oracle Data Integrator (ODI) Oracle Real Application Clusters (RAC) connections, Oracle RAC retry is supported for the ODI master or ODI work repository. ODI uses transactional connections to source and target connections while running ODI scenarios. For these source and target connections, ODI does not support RAC retry connectivity. You cannot migrate these transactions to another node in Oracle RAC.

9.2.2 Configuring Repository Connections to Oracle RAC

When you create an ODI repository using the Repository Creation Utility (RCU), you specify the work repository connection JDBC URL. RCU stores the URL in the master repository contents. If the work repository JDBC URL is a single node URL, you should modify the URL to include the Oracle Real Application Clusters (Oracle RAC) failover address.

- If Oracle RAC is not configured with Single Client Access Name (SCAN), you can provide details of the Oracle RAC instances. In the work repository JDBC Url field, enter the Oracle RAC connectivity address in the format *host*:*port*. See Example 9–1.
- If Oracle RAC is configured with SCAN, provide Oracle RAC instance details with the SCAN address.

Example 9–1 shows the JDBC URL format to connect to an Oracle RAC with two hosts when it does not use SCAN:

Example 9–1 JDBC URL Format (Oracle RAC Not Configured with SCAN)

```
jdbc:oracle:thin:(DESCRIPTION = (LOAD_BALANCE=ON) (ADDRESS = (PROTOCOL = tcp)
(HOST = host1)(PORT = port1)) (ADDRESS = (PROTOCOL = tcp)(HOST = host2)
(PORT = port2)) (CONNECT_DATA = (SERVER=dedicated)
(SERVICE_NAME=service_name)))
```

See the topic "Creating a Work Repository" in the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* for more information.