Oracle® Real Application Clusters

Administration and Deployment Guide

11*g* Release 1 (11.1)

B28254-01

July 2007

ORACLE®

Oracle Real Application Clusters Administration and Deployment Guide, 11*g* Release 1 (11.1)

B28254-01

Primary Authors: Mark Bauer, Viv Schupmann

Contributing Authors: Troy Anthony, Lance Ashdown, Prasad Bagal, Anand Beldalker, Eric Belden, George Claborn, Carol Colrain, Jonathan Creighton, Rajesh Dasari, Steve Fogel, Richard Frank, GP Prabhaker Gongloor, Wei Hu, Yong Hu, Rajiv Jayaraman, Dominique Jeunot, Sameer Joshi, Roland Knapp, Raj Kumar, Ken Lee, Karen Li, Barb Lundhild, Venkat Maddali, Bill Manry, Gaurav Manglik, Komal Mangtani, John McHugh, Saar Maoz, Matthew Mckerley, Markus Michalewicz, Anil Nair, Philip Newlan, Michael Nowak, Peter Ogilvie, Muthu Olagappan, Bharat Paliwal, Hanlin Qian, Mark Ramacher, Kevin Reardon, Dipak Saggi, Sudheendra Sampath, Daniel Semler, Ara Shakian, Cathy Shea, Khethavath P. Singh, Bipul Sinha, Qun Song, Kesavan Srinivasan, Alok Srivastava, Janet Stern, Richard Strohm, Juan Tellez, Leo Tominna, Peter Wahl, Tak Wang, Douglas Williams, Mike Zampiceni, Michael Zoll

Contents

Preface ix

Audience ix
Documentation Accessibility ix
Related Documents x
Conventions xi

What's New in Oracle Real Application Clusters Administration and Deployment? xiii

Oracle Database 11g Release 1 (11.1) New Features in Oracle Real Application Clusters xiii

1 Introduction to Oracle Real Application Clusters

Overview of Oracle Real Application Clusters 1-1
Overview of Oracle Clusterware for Oracle Real Application Clusters 1-3
Overview of Oracle Real Application Clusters Architecture and Processing 1-4
Understanding Cluster-Aware Storage Solutions 1-4
Overview of Connecting to the Oracle Database Using Services and VIP Addresses 1-4
About Oracle Real Application Clusters Software Components 1-5
About Oracle Real Application Clusters Background Processes 1-6
Overview of Automatic Workload Management 1-7
Overview of Installing Oracle Real Application Clusters 1-8
Understanding Compatibility in Oracle Real Application Clusters Environments 1-9
Overview of Oracle Real Application Clusters Installation and Database Creation 1-9
Overview of Extending Oracle ASM and Oracle Real Application Clusters Software 1-10
Overview of Managing Oracle Real Application Clusters Environments 1-11
About Designing and Deploying Oracle Real Application Clusters Environments 1-11
About Administrative Tools for Oracle Real Application Clusters Environments 1-11
About Monitoring Oracle Real Application Clusters Environments 1-12
About Evaluating Performance in Oracle Real Application Clusters Environments 1-13

2 Administering Storage

Overview of Storage in Oracle Real Application Clusters 2-1
Oracle Flexible Architecture 2-2
Datafile Access in Oracle Real Application Clusters 2-2
Redo Log File Storage in Oracle Real Application Clusters 2-2
Automatic Undo Management in Oracle Real Application Clusters 2-3

Automatic Storage Management in Oracle Real Application Clusters 2-3
ASM Storage Management in Oracle Real Application Clusters 2-4
Modifying Disk Group Configurations for ASM in Oracle Real Application Clusters 2-4
Standalone ASM Disk Group Management 2-5
Performing Automatic Storage Management Rolling Upgrades 2-5
Configuring Preferred Mirror Read Disks in Extended Distance Clusters 2-5
Converting Single-Instance ASM to Clustered ASM 2-6
Extending ASM to Nodes Running Single-Instance or Oracle RAC Databases 2-6
Administering ASM Instances and Disk Groups with Enterprise Manager in Oracle RAC .. 2-9
Administering ASM Instances with SRVCTL in Oracle Real Application Clusters 2-9

3 Administering Database Instances and Cluster Databases

Tools for Administering Oracle Real Application Clusters 3-1
Administering Oracle Real Application Clusters with Enterprise Manager 3-1
Administering Oracle Real Application Clusters with SQL*Plus 3-2
Administering Oracle Real Application Clusters with SRVCTL 3-4
Starting and Stopping Instances and Oracle Real Application Clusters Databases 3-4
Starting and Stopping with Enterprise Manager 3-5
Starting Up and Shutting Down with SQL*Plus 3-5
Starting Up and Shutting Down with SRVCTL 3-6
Verifying That Instances are Running 3-7
Terminating Sessions On a Specific Cluster Instance 3-7
Overview of Initialization Parameter Files in Oracle Real Application Clusters 3-9
Setting SPFILE Parameter Values for Oracle Real Application Clusters 3-10
Parameter File Search Order in Oracle Real Application Clusters 3-11
Backing Up the Server Parameter File 3-11
Initialization Parameter Use in Real Application Clusters 3-11
Parameters That Must Have Identical Settings on All Instances 3-13
Parameters That Have Unique Settings on All Instances 3-14
Parameters That Should Have Identical Settings on All Instances 3-14
Quiescing Oracle Real Application Clusters Databases 3-15
Administering Multiple Cluster Interconnects on Linux and UNIX Platforms 3-16
Recommendations for Setting the CLUSTER_INTERCONNECTS Parameter 3-16
Usage Examples for the CLUSTER_INTERCONNECTS Parameter 3-17
Customizing How Oracle Clusterware Manages Oracle RAC Databases 3-18
Advanced Oracle Enterprise Manager Administration 3-20
Using Enterprise Manager Grid Control to Discover Nodes and Instances 3-20
Administering Jobs and Alerts in Oracle Real Application Clusters 3-21

4 Introduction to Automatic Workload Management

Overview of Automatic Workload Management 4-1
Automatic Workload Repository 4-3
Service Deployment Options 4-3
Using Oracle Services 4-3
Default Service Connections 4-4
Connection Load Balancing 4-5
Fast Application Notification 4-7

Overview of Fast Application Notification 4-7
Application High Availability with Services and FAN 4-8
Managing Unplanned Outages 4-8
Managing Planned Outages 4-9
Fast Application Notification High Availability Events 4-9
Using Fast Application Notification Callouts 4-10
Load Balancing Advisory 4-10
Overview of the Load Balancing Advisory 4-11
Configuring Your Environment to Use the Load Balancing Advisory 4-11
Load Balancing Advisory FAN Events 4-12
Oracle Clients That Are Integrated with Fast Application Notification 4-12
Enabling JDBC Clients for Fast Connection Failover 4-13
Enabling Oracle Call Interface Clients for Fast Connection Failover 4-15
Enabling Oracle Call Interface Clients for Runtime Connection Load Balancing 4-16
Enabling ODP.NET Clients to Receive FAN High Availability Events 4-16
Enabling ODP.NET Clients to Receive FAN Load Balancing Advisory Events 4-17
Enabling Event Notification for Connection Failures in Oracle Real Application Clusters . 4-18
Services and Distributed Transaction Processing in Oracle Real Application Clusters 4-22
Enabling Distributed Transaction Processing for Services 4-23
Administering Services 4-24
Administering Services with Enterprise Manager, PL/SQL, and SRVCTL 4-26
Administering Services with Enterprise Manager 4-26
Administering Services with the PL/SQL DBMS_SERVICE Package 4-28
Administering Services with SRVCTL 4-29
Measuring Performance by Service Using the Automatic Workload Repository 4-30
Service Thresholds and Alerts 4-32
Example of Services and Thresholds Alerts 4-32
Enable Service, Module, and Action Monitoring 4-33

5 Configuring Recovery Manager and Archiving

Overview of Configuring RMAN for Oracle Real Application Clusters 5-1
Configuring the RMAN Snapshot Control File Location 5-1
Configuring RMAN to Automatically Backup the Control File and SPFILE 5-2
Crosschecking on Multiple Oracle Real Application Clusters Nodes 5-3
Configuring Channels for RMAN in Oracle Real Application Clusters 5-3
Configuring Channels to Use Automatic Load Balancing 5-3
Configuring Channels to Use a Specific Channel 5-3
Managing Archived Redo Logs Using RMAN in Oracle Real Application Clusters 5-4
Archived Redo Log File Conventions in Oracle Real Application Clusters 5-5
RMAN Archiving Configuration Scenarios 5-5
Automatic Storage Management and Cluster File System Archiving Scheme 5-6
Noncluster File System Local Archiving Scheme 5-7
Changing the Archiving Mode in Oracle Real Application Clusters 5-9
Monitoring the Archiver Processes 5-9

6 Managing Backup and Recovery

RMAN Backup Scenario for Noncluster File System Backups 6-1
RMAN Restore Scenarios for Oracle Real Application Clusters 6-1
Cluster File System Restore Scheme 6-2
Noncluster File System Restore Scheme 6-2
Using RMAN or Enterprise Manager to Restore the Server Parameter File (SPFILE) 6-2
Instance Recovery in Oracle Real Application Clusters 6-2
Single Node Failure in Oracle Real Application Clusters 6-3
Multiple-Node Failures in Oracle Real Application Clusters 6-3
Using RMAN to Create Backups in Oracle Real Application Clusters 6-3
Channel Connections to Cluster Instances 6-3
Node Affinity Awareness of Fast Connections 6-4
Deleting Archived Redo Logs after a Successful Backup 6-4
Autolocation for Backup and Restore Commands 6-5
Media Recovery in Oracle Real Application Clusters 6-5
Parallel Recovery in Oracle Real Application Clusters 6-5
Parallel Recovery with RMAN 6-6
Disabling Parallel Recovery 6-6
Using a Flash Recovery Area in Oracle Real Application Clusters 6-6

7 Using Cloning to Add ASM and Oracle RAC to Nodes in a New Cluster

Introduction to Cloning Oracle ASM and Oracle RAC 7-1
Preparing to Clone ASM and Oracle RAC 7-2
Deploying ASM and Oracle RAC to Other Nodes in the Cluster 7-3
Deploying ASM Instance Homes 7-3
Deploying Oracle RAC Database Homes 7-6
Locating and Viewing Log Files Generated During Cloning 7-9

8 Using Cloning to Extend Oracle RAC to Nodes in the Same Cluster

Adding Nodes Using Cloning in Oracle Real Application Clusters Environments 8-1

9 Adding and Deleting Oracle RAC from Nodes on Linux and UNIX Systems

Adding Oracle RAC to Nodes Running Clusterware and Oracle Database 9-1
Adding Oracle RAC to Nodes That Do Not Have Clusterware and Oracle Database 9-2
Prerequisite Steps for Extending Oracle RAC to Target Nodes 9-2
Extend Oracle Clusterware to Target Nodes 9-4
Configure Shared Storage on Target Nodes 9-5
Add the Oracle Real Application Clusters Database Homes to Target Nodes 9-6
Add ASM and Oracle RAC Database Instances to Target Nodes 9-9
Deleting Cluster Nodes from Oracle Real Application Clusters Environments 9-11
Step 1: Delete Instances from Oracle Real Application Clusters Databases 9-11
Step 2: Delete Nodes from the Cluster 9-13

10 Design and Deployment Techniques

Deploying Oracle Real Application Clusters for High Availability 10-1

Best Practices for Deploying Oracle RAC in a High Availability Environment 10-2
Consolidating Multiple Applications in a Database or Multiple Databases in a Cluster..... 10-3
Scalability of Oracle Real Application Clusters 10-4
General Design Considerations for Oracle Real Application Clusters 10-4
General Database Deployment Topics for Oracle Real Application Clusters 10-5
Tablespace Use in Oracle Real Application Clusters 10-5
Object Creation and Performance in Oracle Real Application Clusters 10-6
Node Addition and Deletion and the SYSAUX Tablespace in Oracle RAC..... 10-6
Distributed Transactions and Oracle Real Application Clusters 10-6
Deploying OLTP Applications in Oracle Real Application Clusters 10-6
Flexible Implementation with Cache Fusion 10-7
Deploying Data Warehouse Applications with Oracle Real Application Clusters..... 10-7
Data Security Considerations in Oracle Real Application Clusters..... 10-7

11 Monitoring Performance

Overview of Monitoring and Tuning Oracle Real Application Clusters Databases 11-1
Monitoring Oracle Real Application Clusters and Oracle Clusterware 11-2
Tuning Oracle Real Application Clusters Databases..... 11-4
Verifying the Interconnect Settings for Oracle Real Application Clusters..... 11-5
Influencing Interconnect Processing..... 11-5
Performance Views in Oracle Real Application Clusters..... 11-6
Creating Oracle Real Application Clusters Data Dictionary Views with CATCLUST.SQL .. 11-6
Oracle Real Application Clusters Performance Statistics 11-6
Automatic Workload Repository in Oracle Real Application Clusters Environments 11-7
Active Session History Reports for Oracle RAC..... 11-7
Overview of ASH Reports for Oracle RAC 11-8
ASH Report for Oracle RAC: Top Cluster Events..... 11-8
ASH Report for Oracle RAC: Top Remote Instance..... 11-8
Monitoring Oracle Real Application Clusters Statistics and Wait Events..... 11-8
Oracle Real Application Clusters Statistics and Events in AWR and Statspack Reports..... 11-9
Oracle Real Application Clusters Wait Events 11-9
Monitoring Performance by Analyzing GCS and GES Statistics 11-10
Analyzing Cache Fusion Transfer Impact Using GCS Statistics 11-11
Analyzing Response Times Based on Wait Events 11-11

A Server Control Utility Reference

Overview of SRVCTL for Oracle Real Application Clusters and Oracle Clusterware A-1
Guidelines for Using SRVCTL in Oracle Real Application Clusters A-1
Obtaining Command-Line Help for SRVCTL A-2
SRVCTL Command Syntax and Options A-2
SRVCTL Cluster Database Configuration Tasks A-3
SRVCTL General Cluster Database Administration Tasks A-3
SRVCTL Node-Level Tasks A-3
SRVCTL Command Reference A-4
SRVCTL Commands..... A-4
SRVCTL Commands Summary A-4

SRVCTL Objects Summary A-5
srvctl add A-5
srvctl config A-9
srvctl enable A-11
srvctl disable A-13
srvctl start A-15
srvctl stop A-18
srvctl modify A-21
srvctl relocate A-25
srvctl status A-26
srvctl getenv A-28
srvctl setenv and unsetenv A-30
srvctl remove A-33

B Troubleshooting Oracle Real Application Clusters

Where to Find Files for Analyzing Errors B-1
Managing Diagnostic Data in Oracle Real Application Clusters B-2
Using Instance-Specific Alert Files in Oracle Real Application Clusters B-2
Enabling Tracing for Java-Based Tools and Utilities in Oracle Real Application Clusters B-3
Resolving Pending Shutdown Issues B-3
How to Determine If Oracle RAC Instances Are Using the Private Network B-3

C Oracle Real Application Clusters Tools Messages

Overview of Messages Specific to Oracle Real Application Clusters C-1
Prefixes and Message Codes for Oracle Real Application Clusters Messages C-2
Types of Oracle Real Application Clusters Messages and Related Files C-2
PRKA—Cluster Node Applications Messages C-2
PRKC—Cluster Command Messages C-4
PRKD—Global Services Daemon Messages C-14
PRKE—Global Services Daemon Controller Utility Messages C-14
PRKH—Server Manager (SRVM) Messages C-15
PRKI—Cluster Pre-Install Messages C-16
PRKN—Server Manager (SRVM) System Library Messages C-18
PRKO—Server Control (SRVCTL) Utility Messages C-18
PRKP—Cluster Database Management Messages C-22
PRKR—Cluster Registry Messages C-29
PRKS—Automatic Storage Management Messages C-35
PRKU—Command-Line Parser Utility Messages C-39
PRKV—Virtual IP Configuration Assistant Messages C-39

Index

Preface

The *Oracle Real Application Clusters Administration and Deployment Guide* describes the Oracle Real Application Clusters (Oracle RAC) architecture and provides an overview of this product. This book also describes administrative and deployment topics for Oracle RAC.

Information in this manual applies to Oracle RAC as it runs on all platforms unless otherwise noted. In addition, the content of this manual supplements administrative and deployment topics for Oracle single-instance databases that appear in other Oracle documentation. Where necessary, this manual refers to platform-specific documentation. This Preface contains these topics:

- Audience
- Documentation Accessibility
- Related Documents
- Conventions

Audience

The *Oracle Real Application Clusters Administration and Deployment Guide* is intended for database administrators, network administrators, and system administrators who perform the following tasks:

- Install and configure an Oracle RAC database
- Administer and manage Oracle RAC databases
- Manage and troubleshoot clusters and networks that use Oracle RAC

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at

`http://www.oracle.com/accessibility/`

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

TTY Access to Oracle Support Services

Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, seven days a week. For TTY support, call 800.446.2398.

Related Documents

This book, the *Oracle Real Application Clusters Administration and Deployment Guide*, provides administration and application deployment information that is specific to Oracle RAC. The discussions herein assume a knowledge of Oracle Clusterware.

For more information, see the Oracle resources listed in this section.

- *Oracle Database 2 Day + Real Application Clusters Guide*

  This task-oriented guide helps you understand the basic steps required to install, configure, and administer Oracle Clusterware and Oracle Real Application Clusters on a two-node system using Red Hat Linux system.

- *Oracle Clusterware Administration and Deployment Guide*

  This is an essential companion book that describes Oracle Clusterware components such as the voting disks and the Oracle Cluster Registry (OCR).

- Platform-specific Oracle Clusterware and Oracle Real Application Clusters installation guides

  Each platform-specific Oracle Database installation media contains a copy of an Oracle Clusterware and Oracle RAC platform-specific installation and configuration guide in HTML and PDF formats. These installation books contain the preinstallation, installation, and post-installation information for the various UNIX, Linux, and Windows platforms on which Oracle Clusterware and Oracle RAC operate.

- *Oracle Database 2 Day DBA*
- *Oracle Database Administrator's Guide*
- *Oracle Database Net Services Administrator's Guide*
- *Oracle Database Platform Guide for Microsoft Windows*
- *Oracle Database 11g Administrator's Reference Release 1 (11.1) for UNIX Systems: AIX Systems, HP-UX, Linux, and the Solaris Operating System (SPARC)*

Note: Additional information for this release may be available in the Oracle Database 11*g* README or Release Notes. If these documents are available for this release, then they are on your Oracle product installation media.

Database error messages descriptions are available online or by way of a Tahiti documentation search. Java tools messages that are specific to Oracle RAC appear in Appendix C of this document.

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.

What's New in Oracle Real Application Clusters Administration and Deployment?

This section describes the new administration and deployment features for Oracle Real Application Clusters (Oracle RAC) in Oracle Database 11*g* Release 1 (11.1).

Oracle Database 11g Release 1 (11.1) New Features in Oracle Real Application Clusters

This section describes the Oracle Database 11*g* release 1 (11.1) features for Oracle RAC administration and deployment.

- **Oracle Clusterware administration content moved to a separate book**

  Oracle Clusterware administration is now documented in a separate book.

  > **See Also:** Your platform-specific Oracle Clusterware installation guide for more information about installing Oracle Clusterware, your platform-specific Oracle RAC installation guide for information about installing Oracle RAC, and *Oracle Clusterware Administration and Deployment Guide* for more information about Oracle Clusterware administration

- **Oracle cloning procedures for ASM and Oracle RAC Databases**

  A new chapter provides step-by-step procedures for cloning Oracle Automatic Storage Management (ASM) and Oracle RAC homes to quickly extend ASM and Oracle RAC environments to additional nodes with the same configuration.

  > **See Also:** Chapter 7, "Using Cloning to Add ASM and Oracle RAC to Nodes in a New Cluster"

- **Parallel execution processes run on database instances according to service placement**

  Parallel execution is now aware of the service definition and automatically takes on the appropriate `PARALLEL_INSTANCE_GROUPS` setting, making it unnecessary to explicitly set the `PARALLEL_INSTANCE_GROUPS` initialization parameter for an Oracle RAC database. Thus, if you execute a SQL statement in parallel, then the default behavior is for Oracle to run the parallel processes only on the instances that offer the service that you used to connect to the database.

See Also: The topics listed under the heading "Administering Services" on page 4-24 for more information about parallel execution processing in Oracle RAC

- **Extending ASM to Nodes Running Single-Instance or Oracle RAC Databases**

  Using the Database Configuration Assistant (DBCA), you can now add a new ASM instance to a node that is running either a single-instance database or an Oracle RAC database instance.

  See Also: Extending ASM to Nodes Running Single-Instance or Oracle RAC Databases on page 2-6

- **OCI Runtime Connection Load Balancing**

  OCI session pools have a new feature called Runtime Connection Load Balancing. OCI session pools are now integrated with the Oracle RAC load balancing advisory to provide load balancing at the time the application gets the connection from the session pool.

  See Also: "Enabling Oracle Call Interface Clients for Runtime Connection Load Balancing" on page 4-16, and in the *Oracle Call Interface Programmer's Guide*

- **Enhanced support for XA transactions in Oracle RAC environments**

  An XA transaction can now span Oracle RAC instances, allowing any application that uses XA to take full advantage of the Oracle RAC environment. This feature allows the units of work performed across these Oracle RAC instances to share resources and act as a single transaction.

  See Also: "Services and Distributed Transaction Processing in Oracle Real Application Clusters" on page 4-22 and the *Oracle Database Advanced Application Developer's Guide*

- **Automatic Database Diagnostics Monitor (ADDM) enhancements for Oracle RAC and Enterprise Manager support**

  ADDM has been enhanced to provide comprehensive clusterwide performance diagnostic and tuning advice. This enhanced mode of ADDM, called ADDM for Oracle Real Application Clusters, analyzes an Oracle RAC database cluster and reports on issues that are affecting the entire cluster as well as those affecting individual instances.

  This feature is particularly helpful in tuning global resources such as I/O and interconnect traffic, making Oracle RAC database management easier and more precise.

  See Also: Chapter 11, "Monitoring Performance" and the *Oracle Database 2 Day + Real Application Clusters Guide* for more information about how to navigate the ADDM pages in Oracle Enterprise Manager to obtain cluster-wide views

  In addition, Oracle Enterprise Manager supports ADDM enhancements for Oracle RAC:

  – ADDM database-wide analysis for Oracle RAC.

– Targeted ADDM analysis is available for several granularity levels, such as the cluster database, database instance, or for specific targets such as SQL, sessions, services, modules, actions, clients, or wait classes.

See Also: *Oracle Database 2 Day + Performance Tuning Guide*, *Oracle Database 2 Day + Real Application Clusters Guide*, and the Oracle Enterprise Manager online help system

- **ASM preferred mirror read**

  This feature is useful in extended clusters where remote nodes have asymmetric access with respect to performance. This leads to better storage utilization and lower network loading.

  ASM in Oracle Database 10*g* always reads the primary copy of a mirrored extent set. In Oracle Database 11*g*, when you configure ASM failure groups it might be more efficient for a node to read from a failure group extent that is closest to the node, even if it is a secondary extent. You can do this by configuring preferred read failure groups.

  See Also: "Configuring Preferred Mirror Read Disks in Extended Distance Clusters" on page 2-5

- **New advanced fault diagnosability infrastructure**

  The advanced fault diagnosability infrastructure correlates the trace files from across multiple instances. It is collects and manages diagnostic data using the Automatic Diagnostic Repository (ADR) file-based repository, and it uses the ADRCI command-line tool to correlate diagnostics across all instances. Also, you can view the ADR locations by querying the new `GV$DIAG_INFO` view.

  See Also: "Managing Diagnostic Data in Oracle Real Application Clusters" on page B-2

- **New SYSASM Privilege and OSASM operating system group for ASM Administration**

  This feature introduces a new SYSASM privilege that is specifically intended for performing ASM administration tasks. Using the SYSASM privilege instead of the SYSDBA privilege provides a clearer division of responsibility between ASM administration and database administration.

  OSASM is a new operating system group that is used exclusively for ASM. Members of the OSASM group can connect as SYSASM using operating system authentication and have full access to ASM.

  See Also: *Oracle Database Storage Administrator's Guide* for complete information about the SYSASM privilege

- **ASM supports rolling upgrades**

  You can perform rolling upgrades of your ASM software on ASM instances in Oracle RAC. This feature enables you to operate with mixed ASM versions starting with Oracle Database 11*g* release 1 (11.1) and later. As a result, ASM nodes can be independently upgraded or patched without affecting availability of the Oracle RAC database.

See Also: "Performing Automatic Storage Management Rolling Upgrades" beginning on page 2-5

- **ASM manageability features**

  The new storage administration features for ASM manageability include the following:

  - **New attributes for disk group compatibility:** To enable some of the new ASM features, you can use two new disk group compatibility attributes, `COMPATIBLE.RDBMS` and `COMPATIBLE.ASM`. These attributes specify the minimum software version that is required to use disk groups for the database and for ASM respectively. This feature enables heterogeneous environments with disk groups from both Oracle Database 10*g* and Oracle Database 11*g*. By default, both attributes are set to 10.1. You must advance these attributes to take advantage of the new features.
  - **New ASM command-line utility (ASMCMD) commands and options**: ASCMCMD allows ASM disk identification, disk bad block repair, and backup and restore operations in your ASM environment for faster recovery.
  - **ASM fast rebalance:** Rebalance operations that occur while a disk group is in `RESTRICTED` mode eliminate the lock and unlock extent map messaging between ASM instances in Oracle RAC environments, thus improving overall rebalance throughput.

    See Also: *Oracle Database Storage Administrator's Guide* for complete information about ASM

- **Convert single-instance ASM instances to cluster ASM Instances with Enterprise Manager and the rconfig utility**

  You can convert an existing single-instance ASM storage manager to a cluster storage manager, which is required for Oracle RAC databases. You can perform the conversion using either the `rconfig` command-line utility or Oracle Enterprise Manager Grid Control.

  See Also: "Converting Single-Instance ASM to Clustered ASM" on page 2-6

- **Oracle Enterprise Manager Database Control is enhanced to provide improved ASM manageability**

  Oracle Enterprise Manager Database Control enables you to more easily manage several ASM tasks, such as managing disk resynchronization, controlling preferred read settings, and managing rolling upgrades.

  See Also: *Oracle Database Storage Administrator's Guide* for more information about using Enterprise Manager Database Control to manage ASM and "Administering ASM Instances and Disk Groups with Enterprise Manager in Oracle RAC" on page 2-9

- **Optimal Flexible Architecture ensures reliable installations and improves software manageability**

  This feature improves manageability by making default Oracle Database installations more compliant with Optimal Flexible Architecture (OFA) specifications. As a part of this feature, the Oracle Universal Installer has been

updated so that the default installation follows Oracle's Optimal Flexible Architecture. This ensures reliable installations and improves software manageability.

See Also: Chapter 2, "Administering Storage"

- **Ability to kill sessions from any instance in a cluster**

  The new `SESSION` parameter on the `ALTER SYSTEM KILL` statement enables you to terminate a session on a specific instance.

  See Also: Chapter 3, "Administering Database Instances and Cluster Databases" for more information about terminating sessions

- **Enhanced Oracle Clusterware and Oracle RAC diagnosability, more background processes**

  This feature provides the ability to run cluster-wide manageability and diagnosability commands to perform health checks, change debugging levels, start and stop the Oracle Clusterware, query the active software version, or show an active nodes list.

- **`DBMS_SCHEDULER` package executes and manages jobs in a clustered environment**

  Oracle Database provides advanced job scheduling capabilities through Oracle Scheduler (the Scheduler). The Scheduler fully supports execution of jobs in a clustered environment. To balance the load on your system and for better performance, you can also specify the database service where you want a job to run.

  See Also: The "Using the Scheduler in Real Application Clusters Environments" section in *Oracle Database Administrator's Guide* and "Load Balancing Advisory" on page 4-10

- **Enhanced initialization parameter administration**

  This feature improves database manageability by making it easier to administer the server parameter file (SPFILE). A number of enhancements are being introduced to simplify server parameter file management. Some of these enhancements are:

  – A more fault tolerant SPFILE
  – Simplified recovery from the loss of an SPFILE
  – Redesigned, more intuitive, Enterprise Manager initialization parameter management interface
  – Prevention of invalid parameter value settings in SPFILE
  – Simplified changing of list parameters

  See Also: "Overview of Initialization Parameter Files in Oracle Real Application Clusters" on page 3-9

- **Improved Oracle RAC performance monitoring and diagnostics in Oracle Enterprise Manager**

  Both Oracle Enterprise Manager Database Control and Oracle Enterprise Manager Grid Control are cluster aware and provide a central console to manage your

cluster database. From any location where you can access a web browser, you can manage Oracle RAC databases, Oracle Clusterware, application servers, host computers, and Web applications, as well as related hardware and software.

You now have the ability to see any given metric across database instances or hosts in the cluster as a tile chart. This high-level view capability means that you do not have to access each individual database instance for details if you just want to see inclusive, aggregated information. You can review issues that are affecting the entire cluster as well as those that are affecting individual instances. This is a major enhancement in terms of how metrics are monitored for Oracle RAC and Oracle Clusterware. With Oracle 11*g*, you can see the roll-up or summary-based views as well as tile based views if you want to monitor how a metric performs across different instances (hosts) over a period of time.

Also, the Oracle Enterprise Manager Cluster Database Performance page provides a quick glimpse of the performance statistics for a database. Statistics are rolled up across all the instances in the cluster database in charts, and provides a cause and recommended solution for performance problems. There is also information available according to the following sorting methods:

- Aggregate by Waits

  All activity data is presented in categories: CPU, Scheduler, User I/O, System I/O, Concurrency, Application, Commit, Configuration, Administrative, Network, Cluster and Other. Enterprise Manager rolls up the presented data from all of the running Oracle RAC instances.

- Aggregate by Services

  Enterprise Manager rolls up all of the activity data for each service. When the activity data is presented in this way, it is simpler to identify which service is most active and thus needs more analysis.

- Aggregate by Instances

  Enterprise Manager rolls up activity data for each instance.

  See Also: "Overview of Monitoring and Tuning Oracle Real Application Clusters Databases" on page 11-1 and *Oracle Database 2 Day + Real Application Clusters Guide*

- **Database Configuration Assistant (DBCA) no longer manages services**

  Cluster managed services are no longer managed through DBCA. The best practice for managing services is to use the Cluster Managed Services page in Oracle Enterprise Manager Database Control, which is accessible from the Cluster Database Availability Page.

  See Also: *Oracle Database 2 Day + Real Application Clusters Guide*

- **Parallel instance groups deprecated**

  Instance groups are deprecated in Oracle Database Release 11*g* (11.1). The replacement mechanism for limiting the number of available instances is to use Services, as described in "Administering Services" on page 4-24.

1

Introduction to Oracle Real Application Clusters

This chapter introduces Oracle Real Application Clusters (Oracle RAC) and describes how to install, administer, and deploy Oracle RAC.

This chapter includes the following topics:

- Overview of Oracle Real Application Clusters
- Overview of Oracle Clusterware for Oracle Real Application Clusters
- Overview of Oracle Real Application Clusters Architecture and Processing
- Overview of Automatic Workload Management
- Overview of Installing Oracle Real Application Clusters
- Overview of Managing Oracle Real Application Clusters Environments

Overview of 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 Real Application Clusters (Oracle RAC) enables you to cluster Oracle databases. Oracle RAC uses Oracle Clusterware for the infrastructure to bind multiple servers so they operate as a single system.

Oracle Clusterware is a portable cluster management solution that is integrated with the Oracle database. Oracle Clusterware is also a required component for using Oracle RAC. In addition, Oracle Clusterware enables both single-instance Oracle databases and Oracle RAC databases to use the Oracle high-availability infrastructure. Oracle Clusterware enables you to create a clustered pool of storage to be used by any combination of single-instance and Oracle RAC databases.

Oracle Clusterware is the only clusterware that you need for most platforms on which Oracle RAC operates. You can also use clusterware from other vendors if the clusterware is certified for Oracle RAC.

> **See Also:** *Oracle Clusterware Administration and Deployment Guide* and the Oracle Clusterware install guides for more details

Single-instance Oracle databases have a one-to-one relationship between the Oracle database and the instance. Oracle RAC environments, however, have a one-to-many relationship between the database and instances. An Oracle RAC database can have up to 100 instances,[1] all of which access one database. All database instances must use the same interconnect, which can also be used by Oracle Clusterware.

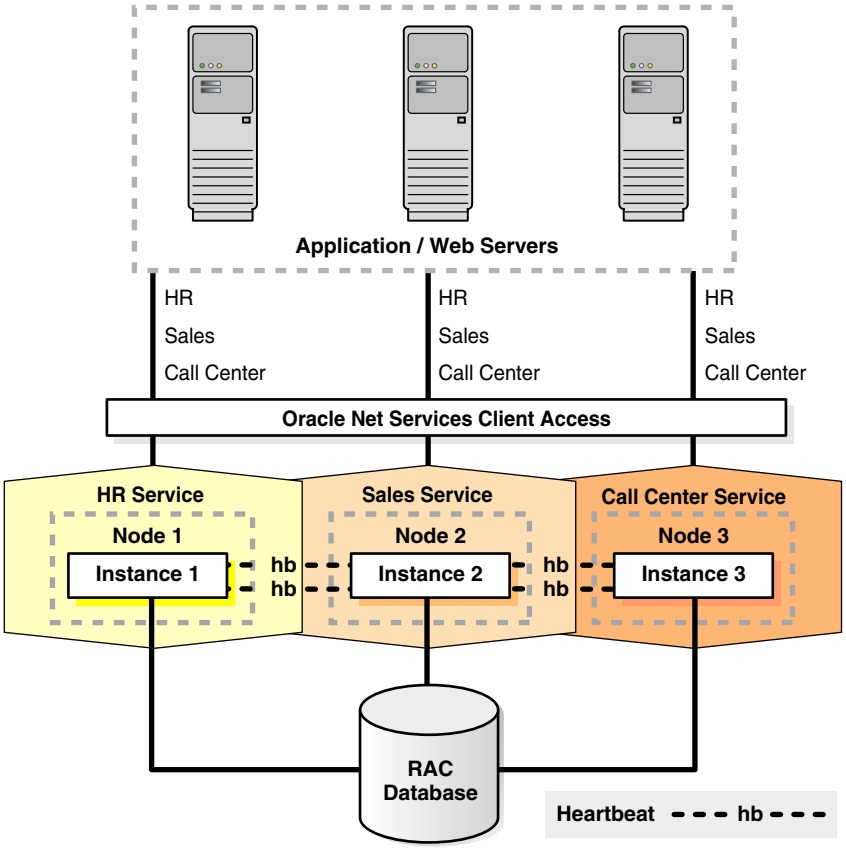
Oracle RAC databases differ architecturally from single-instance Oracle databases in that each Oracle RAC database instance also has:

- At least one additional thread of redo for each instance
- An instance-specific undo tablespace

The combined processing power of the multiple servers can provide greater throughput and Oracle RAC scalability than is available from a single server.

Figure 1–1 shows how Oracle RAC is the Oracle Database option that provides a single system image for multiple servers to access one Oracle database. In Oracle RAC, each Oracle instance usually runs on a separate server.

Figure 1–1 Oracle Database with Oracle RAC Architecture



Traditionally, an Oracle RAC environment is located in one data center. However, you can configure Oracle RAC on an extended distance cluster, which is an architecture that provides extremely fast recovery from a site failure and allows for all nodes, at all sites, to actively process transactions as part of single database cluster. In an extended distance cluster, the nodes in the cluster are located in two buildings that are separated by greater distances (anywhere from across the street, to across a campus, or across a city). For availability reasons, the data needs to be located at both sites, thus requiring the need to implement disk mirroring technology for storage.

If you choose to implement this architecture, you must assess whether this architecture is a good solution for your business, especially with regard to distance, latency, and

[1] For configurations running Oracle Database 10*g* Release 2 and later releases, Oracle Clusterware supports 100 nodes in a cluster, and Oracle RAC supports 100 instances in an Oracle RAC database.

the degree of protection it provides. Oracle RAC on extended distance clusters provides higher availability than is possible with a local Oracle RAC configurations, but an extended distance cluster may not fulfill all of the disaster-recovery requirements of your organization. A feasible separation provides great protection for some disasters (for example, local power outage, airplane crash, server room flooding) but it cannot provide protection against all types of outages. For comprehensive protection against disasters—including protection against corruptions and regional disasters—Oracle recommends the use of Oracle Data Guard with Oracle RAC, as described in the *Oracle Database High Availability Overview* and on the Maximum Availability Architecture (MAA) Web site at

`http://www.oracle.com/technology/deploy/availability/htdocs/maa.htm`

Oracle RAC is a unique technology that provides high availability and scalability for all application types. The Oracle RAC infrastructure is also a key component for implementing the Oracle enterprise grid computing architecture. Having multiple instances access a single database prevents the server from being a single point of failure. Oracle RAC enables you to combine smaller commodity servers into a cluster to create scalable environments that support mission critical business applications. Applications that you deploy on Oracle RAC databases can operate without code changes.

Overview of Oracle Clusterware for Oracle Real Application Clusters

Oracle Clusterware provides a complete, integrated clusterware management solution on all Oracle Database platforms. This clusterware functionality provides all of the features required to manage your cluster database including node membership, group services, global resource management, and high availability functions.

You can install Oracle Clusterware independently or as a prerequisite to the Oracle RAC installation process. Oracle database features such as services use the underlying Oracle Clusterware mechanisms to provide their capabilities. Oracle also continues to support select third-party clusterware products on specified platforms.

Oracle Clusterware is designed for, and tightly integrated with, Oracle RAC. When you create an Oracle RAC database using any of the management tools, the database is registered with and managed by Oracle Clusterware, along with the other Oracle processes such as Virtual Internet Protocol (VIP) address, Global Services Daemon (GSD), the Oracle Notification Service (ONS), and the Oracle Net listeners. These resources are automatically started when Oracle Clusterware starts the node and automatically restarted if they fail. The Oracle Clusterware daemons run on each node.

You can use Oracle Clusterware to manage high-availability operations in a cluster. Anything that Oracle Clusterware manages is known as a CRS resource, which could be a database, an instance, a service, a listener, a VIP address, an application process, and so on. Oracle Clusterware manages CRS resources based on the resource's configuration information that is stored in the Oracle Cluster Registry (OCR). You can use `SRVCTL` commands to administer other node resources. Oracle Clusterware stores the information that describes the configuration of these components in the OCR that you can administer as described in the *Oracle Clusterware Administration and Deployment Guide*.

Overview of Oracle Real Application Clusters Architecture and Processing

At a minimum, Oracle RAC requires a cluster software infrastructure that can provide concurrent access to the same storage and the same set of data files from all nodes in the cluster, a communications protocol for enabling interprocess communication (IPC) across the nodes in the cluster, enable multiple database instances to process data as if the data resided on a logically combined, single cache, and a mechanism for monitoring and communicating the status of the nodes in the cluster.

The following sections describe these concepts in more detail:

- Understanding Cluster-Aware Storage Solutions
- Overview of Connecting to the Oracle Database Using Services and VIP Addresses
- About Oracle Real Application Clusters Software Components
- About Oracle Real Application Clusters Background Processes

Understanding Cluster-Aware Storage Solutions

An Oracle RAC database is a **shared everything** database. All data files, control files, SPFILEs,[2] and redo log files in Oracle RAC environments must reside on cluster-aware shared disks so that all of the cluster database instances can access these storage components. All database instances must use the same interconnect, which can also be used by Oracle Clusterware. Because Oracle RAC databases use a shared everything architecture, Oracle RAC requires cluster-aware storage for all database files.

In Oracle RAC, the Oracle Database software manages disk access and the Oracle software is certified for use on a variety of storage architectures. It is your choice as to how to configure your disk, but you must use a supported cluster-aware storage solution. Oracle Database provides the following file storage options for Oracle RAC:

- Automatic Storage Management (ASM)

  This is the recommended solution to manage your disk.

- Oracle Cluster File System (OCFS)

  OCFS is available for Linux and Windows platforms. However you may optionally use a third-party cluster file system or cluster-aware volume manager that is certified for Oracle RAC.

- A network file system
- Raw devices

Overview of Connecting to the Oracle Database Using Services and VIP Addresses

All **node**s in an Oracle RAC environment must connect to a Local Area Network (LAN) to enable users and applications to access the database. Applications should use the Oracle Database services feature to connect to an Oracle database. Services enable you to define rules and characteristics to control how users and applications connect to database instances. These characteristics include a unique name, workload balancing and failover options, and high availability characteristics. Oracle Net Services enable the load balancing of application connections across all of the instances in an Oracle RAC database.

[2] Note that PFILE files do not need to be shared.

Users can access an Oracle RAC database using a client/server configuration or through one or more middle tiers, with or without connection pooling. Users can be database administrators, developers, application users, power users, such as data miners who create their own searches, and so on.

Most public networks typically use TCP/IP, but you can use any supported hardware and software combination. Oracle RAC database instances can be accessed through a database's default IP address and through VIP addresses.

The interconnect network is a private network that connects all of the servers in the cluster. The interconnect network uses a switch (or multiple switches) that only the nodes in the cluster can access. Configure User Datagram Protocol (UDP) on a Gigabit Ethernet for your cluster interconnect. On Linux and Unix systems, you can configure Oracle Clusterware to use either the UDP or Reliable Data Socket (RDS) protocols. Windows clusters use the TCP protocol. Crossover cables are not supported for use with Oracle Clusterware interconnects.

Note: Do not use the interconnect for the private network for user communication, because **Cache Fusion** uses the private interconnect for interinstance communication.

In addition to the node's host name and IP address, you must also assign a virtual host name and an IP address to each node. You should use the virtual host name or VIP address to connect to the database instance. For example, you might enter the virtual host name `CRM` in the address list of the `tnsnames.ora` file.

A virtual IP address is an alternate public address that client connections use instead of the standard public IP address. To configure VIP addresses, you need to reserve a spare IP address for each node, and the IP addresses must use the same subnet as the public network.

If a node fails, then the node's VIP address fails over to another node on which the VIP address can accept TCP connections but it cannot accept Oracle connections. Generally, VIP addresses fail over when:

- The node on which a VIP address runs fails
- All interfaces for the VIP address fail
- All interfaces for the VIP address are disconnected from the network

Clients that attempt to connect to the VIP address receive a rapid `connection refused` error instead of waiting for TCP connect timeout messages. You configure VIP addresses in the address list for your database connection definition to enable connectivity.

If you use Network Attached Storage (NAS), then you are required to configure a second private network. Access to this network is typically controlled by the vendor's software. The private network uses static IP addresses.

About Oracle Real Application Clusters Software Components

Oracle RAC databases have two or more database instances that each contain memory structures and background processes. An Oracle RAC database has the same processes and memory structures as a single-instance Oracle database as well as additional process and memory structures that are specific to Oracle RAC. Any one instance's database view is nearly identical to any other instance's view in the same Oracle RAC database; the view is a single system image of the environment.

Each instance has a buffer cache in its System Global Area (SGA). Using Cache Fusion, Oracle RAC environments logically combine each instance's buffer cache to enable the instances to process data as if the data resided on a logically combined, single cache.

> **Note:** The SGA size requirements for Oracle RAC are greater than the SGA requirements for single-instance Oracle databases due to Cache Fusion.

To ensure that each Oracle RAC database instance obtains the block that it needs to satisfy a query or transaction, Oracle RAC instances use two processes, the Global Cache Service (GCS) and the Global Enqueue Service (GES). The GCS and GES maintain records of the statuses of each data file and each cached block using a Global Resource Directory (GRD). The GRD contents are distributed across all of the active instances, which effectively increases the size of the SGA for an Oracle RAC instance.

After one instance caches data, any other instance within the same cluster database can acquire a block image from another instance in the same database faster than by reading the block from disk. Therefore, Cache Fusion moves current blocks between instances rather than re-reading the blocks from disk. When a consistent block is needed or a changed block is required on another instance, Cache Fusion transfers the block image directly between the affected instances. Oracle RAC uses the private interconnect for interinstance communication and block transfers. The GES Monitor and the Instance Enqueue Process manages access to Cache Fusion resources and enqueue recovery processing.

About Oracle Real Application Clusters Background Processes

The GCS and GES processes, and the GRD collaborate to enable Cache Fusion. The Oracle RAC processes and their identifiers are as follows:

- `ACMS`—Atomic Controlfile to Memory Service (ACMS)

  In an Oracle RAC environment, the atomic controlfile to memory service (ACMS) per-instance process is an agent that contributes to ensuring a distributed SGA memory update is either globally committed on success or globally aborted in the event of a failure.

- `GTX0-j`—Global Transaction Process

  The GTX0-j process provides transparent support for XA global transactions in a RAC environment. The database autotunes the number of these processes based on the workload of XA global transactions.

- `LMON`—Global Enqueue Service Monitor

  The LMON process monitors global enqueues and resources across the cluster and performs global enqueue recovery operations.

- `LMD`—Global Enqueue Service Daemon

  The LMD process manages incoming remote resource requests within each instance.

- `LMS`—Global Cache Service Process

  The LMS process maintains records of the datafile statuses and each cached block by recording information in a Global Resource Directory (GRD). The LMS process also controls the flow of messages to remote instances and manages global data

block access and transmits block images between the buffer caches of different instances. This processing is part of the Cache Fusion feature.

- `LCK0`—Instance Enqueue Process

  The LCK0 process manages non-Cache Fusion resource requests such as library and row cache requests.

- `RMS`*`n`*—Oracle RAC Management Processes (RMS*n*)

  The RMS*n* processes perform manageability tasks for Oracle RAC. Tasks accomplished by an RMS*n* process include creation of resources related Oracle RAC when new instances are added to the clusters.

- RSMN—Remote Slave Monitor manages background slave process creation and communication on remote instances. These background slave processes perform tasks on behalf of a coordinating process running in another instance.

Note: Many of the Oracle components that this section describes are in addition to the components that are described for single-instance Oracle databases in *Oracle Database Concepts.*

Overview of Automatic Workload Management

Automatic workload management enables you to manage the distribution of workloads to provide optimal performance for users and applications. This includes providing the highest availability for database connections, rapid failure recovery, and balancing workloads optimally across the active configuration. Oracle Database with Oracle RAC includes many features that can enhance automatic workload management, such as connection load balancing, fast connection failover, the load balancing advisory, and runtime connection load balancing. Automatic workload management provides the greatest benefits to Oracle RAC environments. You can, however, take advantage of automatic workload management by using Oracle services in single-instance Oracle Databases, especially those that use Data Guard or Streams. Automatic workload management comprises the following components:

- **High Availability Framework**—The Oracle RAC high availability framework enables the Oracle Database to maintain components in a running state at all times. Oracle high availability implies that Oracle Clusterware monitors and restarts critical components if they stop, unless you override the restart processing. Oracle Clusterware and Oracle RAC also provide alerts to clients when configurations change. This enables clients to immediately react to the changes, enabling application developers to hide outages and reconfigurations from end users. The scope of Oracle high availability spans from the restarting of stopped Oracle processes in an Oracle database instance to failing over the processing of an entire instance to other available instances.

- **Load Balancing Advisory**—This is the ability of the database to provide information to applications about the current service levels being provided by the database and its instances. Applications can take advantage of this information to direct connection requests to the instance that will provide the application request with the best service quality to complete the application's processing. Oracle has integrated its Java Database Connectivity (JDBC) and Oracle Data Provider for .NET (ODP.NET) connection pools to work with the load balancing information. Applications can use the integrated connection pools without programmatic changes.

- **Services**—Oracle Database provides a powerful automatic workload management facility, called services, to enable the enterprise grid vision. Services are entities that you can define in Oracle RAC databases. Services enable you to group database workloads and route the work to the optimal instances that are assigned to process the service. Furthermore, you can use services to define the resources that Oracle assigns to process workloads and to monitor workload resources. Applications that you assign to services transparently acquire the defined automatic workload management characteristics, including high availability and load balancing rules. Many Oracle database features are integrated with services, such as Resource Manager, which enables you to restrict the resources that a service can use within an instance. Some database features are also integrated with Oracle Streams, Advanced Queuing (to achieve queue location transparency), and Oracle Scheduler (to map services to specific job classes).

  In Oracle RAC databases, the service performance rules that you configure control the amount of work that Oracle allocates to each available instance for that service. As you extend your database by adding nodes, applications, components of applications, and so on, you can add more services.

- **Connection Load Balancing**— Oracle Net Services provides connection load balancing for database connections. Connection load balancing occurs when the connection is created. Connections for a given service are balanced across all of the running instances that offer the service. You should define how you want connections to be balanced in the service definition. However, you must still configure Oracle Net Services. When you enable the load balancing advisory, the listener uses the load balancing advisory for connection load balancing.

See Also: Chapter 4, "Introduction to Automatic Workload Management"

Overview of Installing Oracle Real Application Clusters

Install Oracle Clusterware and Oracle Database software using Oracle Universal Installer (OUI), and create your database with Database Configuration Assistant (DBCA). This ensures that your Oracle RAC environment has the optimal network configuration, database structure, and parameter settings for the environment that you selected. As a database administrator, after installation your tasks are to administer your Oracle RAC environment at three levels:

- Instance Administration
- Database Administration
- Cluster Administration

This section introduces the installation processes for Oracle RAC under the following topics:

- Understanding Compatibility in Oracle Real Application Clusters Environments
- Overview of Oracle Real Application Clusters Installation and Database Creation
- Overview of Extending Oracle ASM and Oracle Real Application Clusters Software

---

Note: You must first install Oracle Clusterware before installing Oracle RAC. See *Oracle Clusterware Administration and Deployment Guide* for more information.

Understanding Compatibility in Oracle Real Application Clusters Environments

To run Oracle RAC in configurations with different versions of the database in the same cluster, you must also install clusterware. For example, to run Oracle9*i* and Oracle 10*g* in the same cluster:

- For Oracle RAC nodes running the Oracle9*i* database, you must install an Oracle9*i* cluster:
  - For UNIX the cluster can be HACMP, Serviceguard, Sun Cluster, or Veritas SF
  - For Windows and Linux, the cluster is Oracle Cluster Manager
- If you want to install Oracle RAC running Oracle Database 10*g* or later releases, you must also install Oracle Clusterware. See your platform-specific Oracle Clusterware Installation guide and the *Oracle Clusterware Administration and Deployment Guide* for more information.
- If you are running Oracle RAC 10*g* and Oracle RAC 11*g* in the same cluster, you must be running Oracle Clusterware 11*g* (only)

Oracle requires that you install the Oracle9*i* RAC software first if you are going to run it in a cluster with Oracle RAC 10*g* or Oracle RAC 11*g*.

Note: If you are adding Oracle RAC to servers that are part of a cluster, either migrate to Oracle Clusterware or ensure the clusterware you are running is supported to run with Oracle RAC on release 10*g* or later releases and ensure you have installed the correct options for the two to work together. Oracle strongly recommends that you do not run different cluster software on the same servers unless they are certified to work together.

Overview of Oracle Real Application Clusters Installation and Database Creation

Once you have installed Oracle Clusterware and it is operational, run OUI to install the Oracle database software with Oracle RAC components.

During the installation, OUI runs DBCA to create your Oracle RAC database according to the options that you select. DBCA also runs the Net Configuration Assistant (NETCA) to configure the network for your Oracle RAC environment.

> **See Also:** *Oracle Database Net Services Administrator's Guide* for more information about NETCA

Oracle RAC software is distributed as part of the Oracle Database installation media. By default, the standard Oracle Database software installation process installs the Oracle RAC option when it recognizes that you are performing the installation on a cluster. The OUI installs Oracle RAC into a directory structure, which can be referred to as the Oracle home, which is separate from other Oracle software running on the system. Because OUI is cluster aware, it installs Oracle RAC software on all of the nodes that you defined to be part of the cluster.

Oracle recommends that you select ASM during the installation to simplify storage management; ASM automatically manages the storage of all database files within disk groups. You can also configure services during installation, depending on your processing requirements. If you are using Oracle Database Standard Edition, then you *must* use ASM to store all of the database files.

By default, Oracle creates one service for your environment and the service is for the database. (The default database service is typically identified using the combination of the DB_NAME and DB_DOMAIN initialization parameters: *db_name*.*db_domain*.) The default service is available on all instances in an Oracle RAC environment, unless the database is in restricted mode.

> **Note:** Avoid changing host names after you complete the Oracle Clusterware installation, including adding or deleting domain qualifications. Nodes with changed host names must be deleted from the cluster and added back with the new name.

Overview of Extending Oracle ASM and Oracle Real Application Clusters Software

You can extend Oracle ASM and Oracle RAC in grid environments to additional nodes by copying cloned images of the ASM and Oracle RAC database homes to other nodes in the cluster. Oracle cloning copies images of the software to other nodes that have similar hardware and software. Cloning is best suited to a scenario where you need to quickly extend your Oracle RAC environment to several nodes of the same configuration.

Oracle provides the following methods of extending Oracle Clusterware environments:

- Oracle cloning procedure using cloning scripts
- Oracle Enterprise Manager cloning
- The `addNode.sh` script and OUI cloning

> **Note:** Oracle cloning is not a replacement for Oracle Enterprise Manager cloning that is part of the Provisioning Pack. During Oracle Enterprise Manager cloning, the provisioning process includes a series of steps where details about the home you want to capture, the location you want to deploy to, and various other parameters are collected.

For new installations or if you have to install only one Oracle RAC database, you should use the traditional automated and interactive installation methods, such as OUI, or the Provisioning Pack feature of Oracle Enterprise Manager. If your goal is to add or delete Oracle RAC from nodes in the cluster, you can use the `addNode.sh` and `rootdelete.sh` scripts.

The cloning process assumes that you successfully installed an Oracle Clusterware home and an Oracle home with Oracle RAC on at least one node. In addition, all root scripts must have run successfully on the node from which you are extending your cluster database.

At a high level, Oracle cloning involves the following main tasks:

1. Clone the Oracle Clusterware home following the instructions in *Oracle Clusterware Administration and Deployment Guide*.
2. Clone the Oracle home with the Oracle RAC software.

   The process for cloning the Oracle home onto new nodes is similar to the process for cloning the Oracle Clusterware home.
3. Run NETCA on each new node to create a listener.

4. If you have not already created a database, then run the DBCA to create one.
5. Follow the post-cloning procedures to complete the extension of your Oracle RAC environment onto the new nodes.

See Also:

- Chapter 7, "Using Cloning to Add ASM and Oracle RAC to Nodes in a New Cluster"
- Chapter 9, "Adding and Deleting Oracle RAC from Nodes on Linux and UNIX Systems" for information about adding and deleting nodes and instances on Linux and UNIX Systems
- Oracle Enterprise Manager online Help system for more information about the Provisioning Pack
- *Oracle Database Net Services Administrator's Guide* for more information about NETCA

Overview of Managing Oracle Real Application Clusters Environments

This section describes the following Oracle RAC environment management topics:

- About Designing and Deploying Oracle Real Application Clusters Environments
- About Administrative Tools for Oracle Real Application Clusters Environments
- About Monitoring Oracle Real Application Clusters Environments
- About Evaluating Performance in Oracle Real Application Clusters Environments

About Designing and Deploying Oracle Real Application Clusters Environments

Any enterprise that is designing and implementing a high availability strategy with Oracle RAC must begin by performing a thorough analysis of the business drivers that require high availability. An analysis of business requirements for high availability combined with an understanding of the level of investment required to implement different high availability solutions enables the development of a high availability architecture that will achieve both business and technical objectives. See the following resources for help choosing and implementing the architecture that best fits your availability requirements:

- Chapter 10, "Design and Deployment Techniques" provides a high-level overview you can use to evaluate the high availability requirements of your business.
- *Oracle Database High Availability Overview* describes how to select the most suitable architecture for your organization, describes several high availability architectures, and provides guidelines for choosing the one that best meets your requirements.

About Administrative Tools for Oracle Real Application Clusters Environments

Oracle enables you to administer a cluster database as a single system image through Enterprise Manager, SQL*Plus, or through Oracle RAC command-line interfaces such as Server Control (`SRVCTL`):

- Oracle Enterprise Manager—Enterprise Manager has both the Database Control and Grid Control GUI interfaces for managing both single-instance database and Oracle RAC database environments. Oracle recommends that you use Enterprise Manager to perform administrative tasks whenever feasible.

- Server Control (SRVCTL)—SRVCTL is a command-line interface that you can use to manage an Oracle RAC database from a single point. You can use SRVCTL to start and stop the database and instances and to delete or move instances and services. You can also use SRVCTL to manage configuration information, Oracle Clusterware, and ASM.
- SQL*Plus—SQL*Plus commands operate on the current instance. The current instance can be either the local default instance on which you initiated your SQL*Plus session, or it can be a remote instance to which you connect with Oracle Net Services.
- Cluster Verification Utility (CVU)—CVU is a command-line tool that you can use to verify a range of cluster and Oracle RAC components such as shared storage devices, networking configurations, system requirements, and Oracle Clusterware, as well as operating system groups and users. You can use CVU for preinstallation checks and for post-installation checks of your cluster environment. CVU is especially useful during preinstallation and during installation of Oracle Clusterware and Oracle RAC components. The OUI runs CVU after installing Oracle Clusterware and Oracle Database to verify your environment.

  Install and use CVU before you install Oracle RAC to ensure that your configuration meets the minimum Oracle RAC installation requirements. Also use the CVU for ongoing administrative tasks, such as node addition and node deletion.
- DBCA—DBCA is the recommended method for creating and initially configuring Oracle RAC databases.
- NETCA—Configures the network for your Oracle RAC environment.

See Also:

- Chapter 3, "Administering Database Instances and Cluster Databases" for an introduction to Oracle RAC administration using Oracle Enterprise Manager, SQL*Plus, and the SRVCTL utility
- "Monitoring Oracle Real Application Clusters and Oracle Clusterware" on page 11-2
- Appendix A, "Server Control Utility Reference" for SRVCTL reference information
- *Oracle Clusterware Administration and Deployment Guide* for information about Oracle Clusterware tools such as the OIFCFG tool for allocating and deallocating network interfaces, the OCRCONFIG command-line tool for managing the Oracle Cluster Registry, and the Cluster Verification Utility (CVU)
- *Oracle Database Net Services Administrator's Guide* for more information about NETCA

About Monitoring Oracle Real Application Clusters Environments

Web-based Enterprise Manager Database Control and Grid Control enable you to monitor an Oracle RAC database. The Enterprise Manager Console is a central point of control for the Oracle environment that you access by way of a graphical user interface (GUI). See Monitoring Oracle Real Application Clusters and Oracle Clusterware on page 11-2 and the *Oracle Database 2 Day + Real Application Clusters Guide*, *Oracle Enterprise Manager Concepts*, for detailed information about using Enterprise Manager to monitor Oracle RAC environments.

Also, note the following recommendations about monitoring Oracle RAC environments:

- Use the Enterprise Manager Console to initiate cluster database management tasks.
- Use Enterprise Manager Grid Control to administer multiple Oracle RAC databases.
- Use the global views, or `GV$` views, which are based on `V$` views. The `catclustdb.sql` script creates the `GV$` views. Run this script if you do not create your database with DBCA. Otherwise, DBCA runs this script for you.
- Use the sophisticated management and monitoring features of the Oracle Database Diagnostic and Tuning packs that include the Automatic Database Diagnostic Monitor (ADDM) and AWR.

Note: Although Statspack is available for backward compatibility, Statspack provides reporting only. You must run Statspack at level 7 to collect statistics related to block contention and segment block waits.

See Also: The *Oracle Database Performance Tuning Guide* describes the Oracle automatic features for performance diagnosing and tuning, including ADDM.

About Evaluating Performance in Oracle Real Application Clusters Environments

You do not need to perform special tuning for Oracle RAC; Oracle RAC scales without special configuration changes. If your application performed well on a single-instance Oracle database, then it will perform well in an Oracle RAC environment. Many of the tuning tasks that you would perform on a single-instance Oracle database can also improve Oracle RAC database performance. This is especially true if your environment required scalability across a greater number of CPUs.

Some of the performance features specific to Oracle RAC include:

- Dynamic Resource Allocation
  - Oracle dynamically allocates Cache Fusion resources as needed
  - The dynamic mastering of resources improves performance by keeping resources local to data blocks
- Cache Fusion Enables A Simplified Tuning Methodology
  - You do not have to tune any parameters for Cache Fusion
  - No application-level tuning is necessary
  - You can use a bottom-up tuning approach with virtually no effect on your existing applications
- More Detailed Performance Statistics
  - More views for Oracle RAC performance monitoring
  - Enterprise Manager Database Control and Grid Control are Integrated with Oracle RAC

2

Administering Storage

This chapter describes storage topics, such as Automatic Storage Management (ASM), in an Oracle Real Application Clusters (Oracle RAC) environment.

This chapter includes the following topics:

- Overview of Storage in Oracle Real Application Clusters
- Oracle Flexible Architecture
- Datafile Access in Oracle Real Application Clusters
- Redo Log File Storage in Oracle Real Application Clusters
- Automatic Undo Management in Oracle Real Application Clusters
- Automatic Storage Management in Oracle Real Application Clusters

> **See Also:** *Oracle Clusterware Administration and Deployment Guide*, your platform-specific Oracle Clusterware installation guide, and your Oracle Real Application Clusters installation guide

Overview of Storage in Oracle Real Application Clusters

All datafiles (including an undo tablespace for each instance) and redo log files (at least two for each instance) must reside in an ASM disk group, on a **cluster file system**, or on shared raw devices. In addition, Oracle recommends that you use one shared server parameter file (SPFILE) with instance-specific entries. Alternatively, you can use a local file system to store instance-specific parameter files (PFILEs).

Unless otherwise noted, Oracle storage features such as ASM, Oracle Managed Files (OMF), automatic segment-space management, and so on, function the same in Oracle RAC environments as they do in single-instance Oracle database environments. See *Oracle Database 2 Day DBA*, *Oracle Database Storage Administrator's Guide*, and the *Oracle Database Administrator's Guide* for additional information about these storage features.

If you do not use ASM, if your platform does not support a cluster file system, or if you do not want to use a cluster file system for database file storage, then create additional raw devices as described in your platform-specific Oracle Real Application Clusters installation and configuration guide. However, Oracle recommends that you use ASM for database file storage, as described later in this chapter in the section titled "Automatic Storage Management in Oracle Real Application Clusters" on page 2-3.

The remainder of this section describes the following topics:

- Oracle Flexible Architecture
- Datafile Access in Oracle Real Application Clusters

- Redo Log File Storage in Oracle Real Application Clusters
- Automatic Undo Management in Oracle Real Application Clusters

Note: To create an Oracle RAC database using the Oracle Database Standard Edition, you *must* use ASM for your database storage.

Oracle Flexible Architecture

Optimal Flexible Architecture (OFA) ensures reliable installations and improves software manageability. This feature streamlines the way in which Oracle software installations are organized, thereby simplifying the on-going management of your installations and improves manageability by making default Oracle Database installs more compliant with OFA specifications.

During installation, you are prompted to specify an Oracle base (`ORACLE_BASE`) location, which is owned by the user performing the installation. You can choose an existing `ORACLE_BASE`, or choose another directory location that does not have the structure for an `ORACLE_BASE` directory.

Using the Oracle base directory path helps to facilitate the organization of Oracle installations, and helps to ensure that installations of multiple databases maintain an OFA configuration. During the installation, `ORACLE_BASE` is the only required input, as the `ORACLE_HOME` will be defaulted based on the value chosen for the `ORACLE_BASE`. In addition, Oracle recommends that you set the `ORACLE_BASE` environment variable in addition to `ORACLE_HOME`, when starting databases. Note that `ORACLE_BASE` may become a required environment variable for database startup in a future release.

See Also: Your platform-specific Oracle Real Application Clusters installation guide for more information about specifying an `ORACLE_BASE` directory

Datafile Access in Oracle Real Application Clusters

All Oracle RAC instances must be able to access all datafiles. If a datafile needs to be recovered when the database is opened, then the first Oracle RAC instance to start is the instance that performs the recovery and verifies access to the file. As other instances start, they also verify their access to the datafiles. Similarly, when you add a tablespace or datafile or bring a tablespace or datafile online, all instances verify access to the file or files.

If you add a datafile to a disk that other instances cannot access, then verification fails. Verification also fails if instances access different copies of the same datafile. If verification fails for any instance, then diagnose and fix the problem. Then run the `ALTER SYSTEM CHECK DATAFILES` statement on each instance to verify datafile access.

Redo Log File Storage in Oracle Real Application Clusters

Each instance has its own online redo log groups. Create these redo log groups and establish group members, as described in the *Oracle Database Administrator's Guide*. To add a redo log group to a specific instance, specify the `INSTANCE` clause on the `ALTER DATABASE ADD LOGFILE` statement, as described in the *Oracle Database SQL Language Reference*. If you do not specify the instance when adding the redo log group, the redo log group is added to the instance to which you are currently connected.

Each instance must have at least two groups of redo log files. You must allocate the redo log groups before enabling a new instance with the `ALTER DATABASE ENABLE INSTANCE` *`instance_name`* command. When the current group fills, an instance begins writing to the next log file group. If your database is in `ARCHIVELOG` mode, then each instance must save filled online log groups as archived redo log files that are tracked in the control file.

During database recovery, all enabled instances are checked to see if recovery is needed. If you remove an instance from your Oracle RAC database, you should disable the instance so it is does not have to be checked during database recovery.

Automatic Undo Management in Oracle Real Application Clusters

Oracle automatically manages undo segments within a specific undo tablespace that is assigned to an instance. Only the instance assigned to the undo tablespace can modify the contents of that tablespace. However, all instances can always read all undo blocks throughout the **cluster** environment for consistent read purposes. Also, any instance can update any undo tablespace during transaction recovery, as long as that undo tablespace is not currently used by another instance for undo generation or transaction recovery.

You assign undo tablespaces in your Oracle RAC database by specifying a different value for the `UNDO_TABLESPACE` parameter for each instance in your SPFILE or individual PFILEs. You cannot simultaneously use automatic undo management and manual undo management in an Oracle RAC database. In other words, all instances of an Oracle RAC database must operate in the same undo mode.

> **See Also:** *Oracle Database Administrator's Guide* for detailed information about creating and managing undo tablespaces

Automatic Storage Management in Oracle Real Application Clusters

ASM automatically maximizes performance by managing the storage configuration across the disks that ASM manages. ASM does this by evenly distributing the database files across all of the available storage within your **cluster database** environment. ASM partitions your total disk space requirements into uniformly sized units across all disks in a disk group. ASM can also automatically mirror data to prevent data loss. Because of these features, ASM also significantly reduces your administrative overhead.

To use ASM in Oracle RAC, select ASM as your storage option when you create your database with the Database Configuration Assistant (DBCA). As in single-instance Oracle databases, using ASM in Oracle RAC does not require I/O tuning.

> **Note:** When installing ASM, you should keep the ASM home separate from the database home directory (Oracle home). By using separate home directories, you can upgrade and patch ASM and the Oracle Database software independently, and you can deinstall Oracle Database software without affecting the ASM instance. See the *Oracle Database 2 Day + Real Application Clusters Guide* for complete information.

The following topics describe ASM and ASM administration as follows:

- ASM Storage Management in Oracle Real Application Clusters

- Modifying Disk Group Configurations for ASM in Oracle Real Application Clusters
- Standalone ASM Disk Group Management
- Performing Automatic Storage Management Rolling Upgrades
- Configuring Preferred Mirror Read Disks in Extended Distance Clusters
- Converting Single-Instance ASM to Clustered ASM
- Extending ASM to Nodes Running Single-Instance or Oracle RAC Databases
- Administering ASM Instances and Disk Groups with Enterprise Manager in Oracle RAC
- Administering ASM Instances with SRVCTL in Oracle Real Application Clusters

ASM Storage Management in Oracle Real Application Clusters

When you create your database, Oracle creates one ASM instance on each node in your Oracle RAC environment if one does not already exist. Each ASM instance has either an SPFILE or PFILE type parameter file. Back up the parameter files and the TNS entries for nondefault Oracle Net listeners.

You can create ASM disk groups and configure mirroring for ASM disk groups using DBCA. After your Oracle RAC database is operational, you can administer ASM disk groups with Enterprise Manager.

You configure ASM in a separate standalone *ASM home*. This enables instances for single-instance databases and Oracle RAC databases to share a single ASM instance on a node. You also have the option to upgrade ASM independently of your database upgrades.

The Oracle tools that you use to manage ASM, including DBCA, Database Upgrade Assistant (DBUA), Enterprise Manager, and the silent mode install and upgrade commands, include options to manage ASM instances and disk groups. For example, you can run DBCA to create a new ASM instance or ASM diskgroup independently of creating a database.

When you choose ASM options when performing installation, upgrades, or other operations, the tool you are using may automatically extend ASM to other nodes in your **cluster**. This can include installing ASM software into the same home as on the current node and starting the ASM instance. For example, if you use DBCA to create a database using a new Oracle home, then DBCA attempts to extend ASM to the new Oracle home on all of the nodes you select.

Modifying Disk Group Configurations for ASM in Oracle Real Application Clusters

When you create a disk group for a cluster or add new disks to an existing clustered disk group, prepare the underlying physical storage on shared disks and give the Oracle user permission to read and write to the disk. The shared disk requirement is the only substantial difference between using ASM in an Oracle RAC database compared to using it in a single-instance Oracle database. ASM automatically re-balances the storage load after you add or delete a disk or disk group.

In a cluster, each ASM instance manages its node's metadata updates to the disk groups. In addition, each ASM instance coordinates disk group metadata with other nodes in the cluster. As in single-instance Oracle databases, you can use Enterprise Manager, DBCA, SQL*Plus, and the Server Control Utility (`SRVCTL`) to administer disk groups for ASM in Oracle RAC. The *Oracle Database Storage Administrator's Guide*

explains how to use SQL*Plus to administer ASM instances. The following sections describe how to use the other tools.

Standalone ASM Disk Group Management

When you create a database using DBCA and you select the ASM storage option, DBCA creates the ASM instances for you if they do not already exist. However, you can also use the standalone ASM disk group management feature to create and manage an ASM instance and its associated disk groups independently of creating a new database. You can use Enterprise Manager or DBCA to add disks to a disk group, to mount a disk group or to mount all of the disk groups, or to create ASM instances. Additionally, you can use Enterprise Manager to dismount and drop disk groups or to delete ASM instances.

To create an ASM instance without creating a database with DBCA, select the **Configure Automatic Storage Management** option on the DBCA Database Options page. You can also use this option to add or mount one or more ASM disk groups. The DBCA then displays the Node Selection page on which you can identify the nodes on which you want to create the ASM instance or on which you want to manage disk groups. If necessary, the next page you must complete is the DBCA Create Instance Page on which you add the information about the ASM instance parameter file and `SYS` password and, for Windows systems, the owner of the ASM-related service.

You can also use the ASM Disk Groups page in DBCA for standalone ASM management. That is, you can configure ASM storage separately from database creation. For example, from the ASM Disk Groups page, you can create new disk groups, add disks to existing disk groups, or mount disk groups that are not currently mounted.

Performing Automatic Storage Management Rolling Upgrades

ASM rolling upgrade enables you to upgrade or patch clustered ASM nodes one at a time, without affecting database availability. During a rolling upgrade, you can maintain a functional cluster while one or more of the nodes in the cluster are running different software versions.

Note: An ASM rolling upgrade applies only to clustered ASM instances. You can perform rolling upgrades only in environments with Oracle Database 11*g* release 1 (11.1) and later releases. In other words, you cannot use the rolling upgrade feature to upgrade from Oracle Database 10*g* to Oracle Database 11*g*.

See Also: *Oracle Database Storage Administrator's Guide* for conceptual information about performing ASM rolling upgrades and patching ASM instances, and the *Oracle Database Upgrade Guide* for step-by-step instructions to upgrade an ASM instance with DBUA and to upgrade an ASM instance manually

Configuring Preferred Mirror Read Disks in Extended Distance Clusters

When you configure ASM failure groups, it may be more efficient for a node to read from an extent that is closest to the node, even if that extent is a secondary extent. You can configure ASM to read from a secondary extent if that extent is closer to the node instead of ASM reading from the primary copy which might be farther from the node. Using preferred read failure groups is most beneficial in an extended distance cluster.

To configure this feature, set the `ASM_PREFERRED_READ_FAILURE_GROUPS` initialization parameter to specify a list of failure group names as preferred read disks. Oracle recommends that you configure at least one mirrored extent copy from a disk that is local to a node in an extended cluster. However, a failure group that is preferred for one instance might be remote to another instance in the same Oracle RAC database. The parameter setting for preferred read failure groups is instance specific.

See Also:

- *Oracle Database Storage Administrator's Guide* for complete information about configuring preferred mirror read disks in extended distance clusters
- *Oracle Database Reference* for information about the `ASM_PREFERRED_READ_FAILURE_GROUPS` initialization parameter

Converting Single-Instance ASM to Clustered ASM

You can use either the `rconfig` command or Oracle Enterprise Manager Grid Control to convert an existing ASM instance from a single-instance storage manager to a cluster storage manager. You can convert ASM instances that are running Oracle Database 10*g* release 10.2 (or later) directly to Oracle Database 11*g*.

See Also:

- *Oracle Database 2 Day + Real Application Clusters Guide* for information about using Enterprise Manager Grid Control to convert single-instance ASM to clustered ASM
- *Oracle Database Storage Administrator's Guide* for complete information about configuring preferred mirror read disks in extended distance clusters
- Your platform-specific Oracle Real Application Clusters installation guide for detailed information about converting ASM using the `rconfig` command

Extending ASM to Nodes Running Single-Instance or Oracle RAC Databases

This section describes how to add a new ASM instance to a node that is running either a single-instance database or an Oracle RAC database instance.

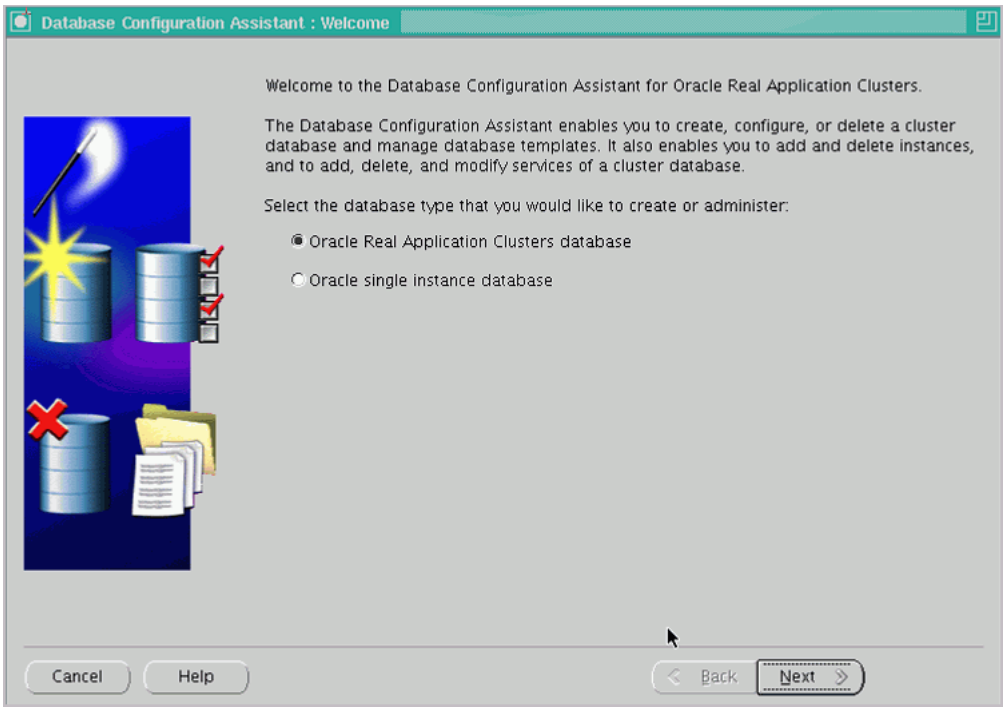
Perform the following steps to extend ASM from an existing node to a new node:

1. Start DBCA from the node where you already have configured the ASM instance. (In this case, the node is called stbdq18).

   You should run the DBCA from the ASM home in the existing cluster not the node you just added. By running DBCA out of the existing ASM home, you ensure that ASM will be running from the correct home.

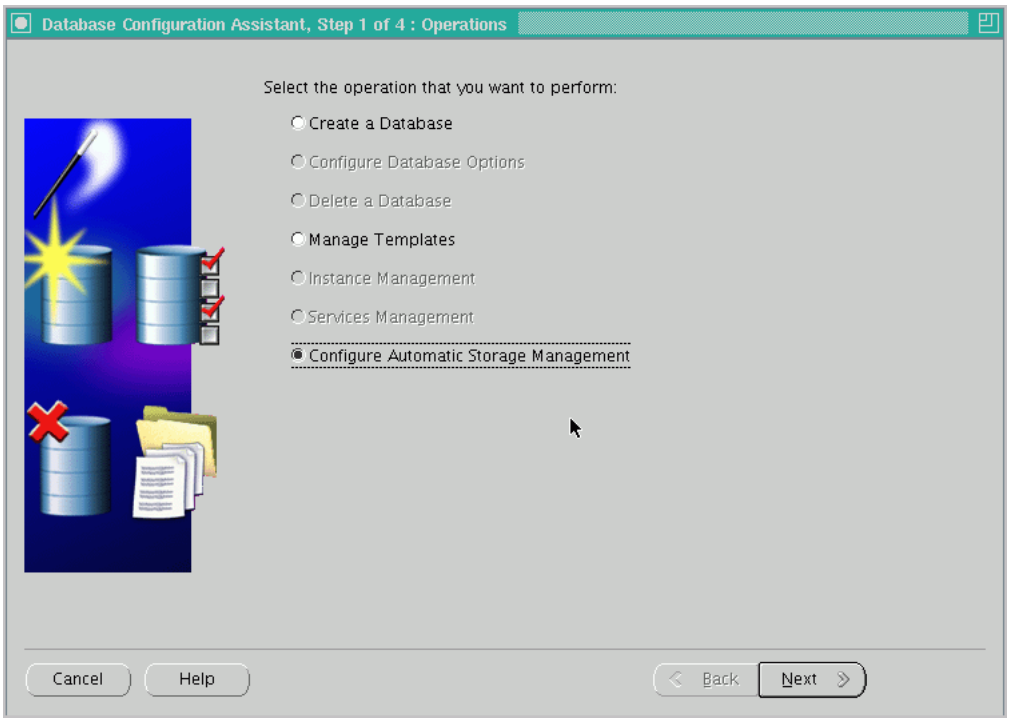
2. On the Welcome screen of the Database Upgrade Assistant (shown in Figure 2–1), select **Oracle Real Application Clusters database**.

Figure 2–1 Extending ASM to a New Node: DBCA Welcome Page



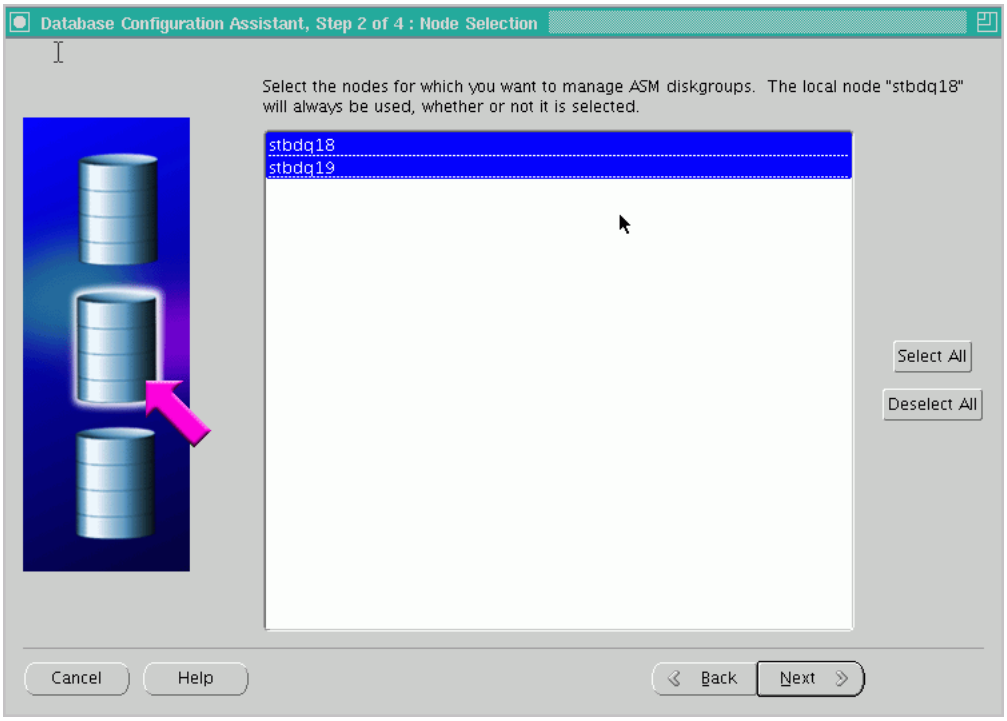
3. On the DBCA Operations screen (shown in Figure 2–2), select the **Configure Automatic Storage Management** option.

Figure 2–2 Extending ASM to a New Node: DBCA Operations Page



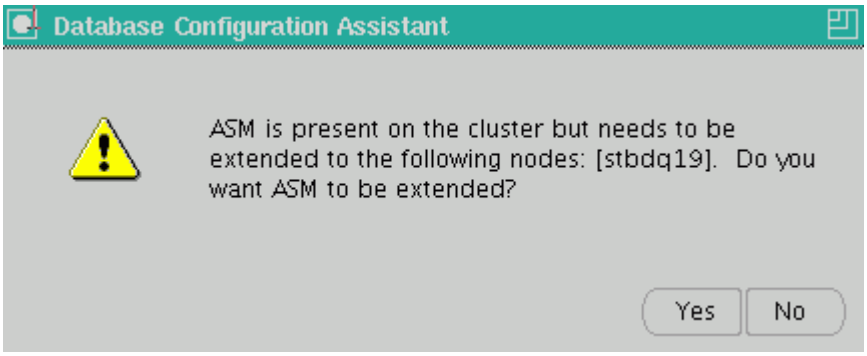
4. On the Node Selection screen (shown in Figure 2–3), you should see both the source node (stbdq18) and the target node (stbdq19). Select both nodes and click **Next**.

Figure 2–3 Extending ASM to a New Node: DBCA Node Selection Screen



5. DBCA checks for ASM availability on the new node provisioned (stbdq19) and displays the confirmation window shown in Figure 2–4.

Figure 2–4 Extending ASM to a New Node: Confirmation Window



6. Select **Yes** to create ASM on the new node. DBCA creates ASM in the new node.
7. You can confirm that ASM has been extended to the new node by issuing the following command:

```
crs_stat | grep asm

NAME=ora.stbdq18.ASM1.asm
NAME=ora.stbdq19.ASM2.asm
```

Administering ASM Instances and Disk Groups with Enterprise Manager in Oracle RAC

You can administer ASM with Oracle Enterprise Manager Database Control (Database Control). Database Control enables you to more easily manage ASM instances, disks, disk groups, and failure groups in Oracle RAC environments.

To begin administering ASM, go to the ASM Home page in Database Control. To access the ASM Home page, you must:

1. Log in to Oracle Enterprise Manager on any node that is running the Oracle Management Service (OMS).

   OMS is automatically started on the node where Database Configuration Assistant (DBCA) was run to create the cluster database. Depending on your configuration, OMS may also be running on other nodes.

2. On the Cluster Database Home page, under the Instances heading, click the link for the desired ASM instance.

You can perform administrative operations on ASM disk groups such as adding and deleting them. You can also monitor ASM disk group performance as well as control disk group availability at the instance level. For example, some of the tasks specific to Oracle RAC and ASM you can perform with Database Control include:

- When you add a disk group, the disk group definition includes a checkbox to indicate whether the disk group is automatically mounted to all of the cluster database instances.
- Monitoring ASM disk group performance—The default Disk Group Performance page displays instance-level performance details when you click a performance characteristic such as **Write Response Time** or **I/O Throughput**.
- When you mount and dismount ASM disk groups, you can use a checkbox to indicate which instances should mount or dismount a particular ASM Disk Group.
- You can manage disk resynchronization, control preferred read settings, and manage ASM rolling upgrades.

> **See Also:** *Oracle Database Storage Administrator's Guide* for complete information about using Database Control to manage ASM in Oracle RAC environments

Administering ASM Instances with SRVCTL in Oracle Real Application Clusters

You can use the Server Control Utility (`SRVCTL`) to add, remove, enable, and disable an ASM instance. To issue `SRVCTL` commands to manage ASM, log in as the operating system user that owns the ASM home and issue the `SRVCTL` commands from the bin directory of the ASM home.

Use the following syntax to add configuration information about an existing ASM instance:

```
srvctl add asm -n node_name -i +asm_instance_name -o oracle_home
```

> **Note:** For all of the `SRVCTL` commands in this section for which the `-i` option is not required, if you do not specify an instance name, then the command applies to all of the ASM instances on the node.

Use the following syntax to remove an ASM instance:

```
srvctl remove asm -n node_name [-i +asm_instance_name]
```

Use the following syntax to enable an ASM instance:

```
srvctl enable asm -n node_name [-i ] +asm_instance_name
```

Use the following syntax to disable an ASM instance:

```
srvctl disable asm -n node_name [-i +asm_instance_name]
```

You can also use SRVCTL to start, stop, and obtain the status of an ASM instance as in the following examples.

Use the following syntax to start an ASM instance:

```
srvctl start asm -n node_name [-i +asm_instance_name] [-o start_options]
```

Use the following syntax to stop an ASM instance:

```
srvctl stop asm -n node_name [-i +asm_instance_name] [-o stop_options]
```

Use the following syntax to show the configuration of an ASM instance:

```
srvctl config asm -n node_name
```

Use the following syntax to obtain the status of an ASM instance:

```
srvctl status asm -n node_name
```

3

Administering Database Instances and Cluster Databases

This chapter describes how to administer Oracle Real Application Clusters (Oracle RAC) database instances and Oracle RAC databases.

The topics in this chapter include:

- Tools for Administering Oracle Real Application Clusters
- Starting and Stopping Instances and Oracle Real Application Clusters Databases
- Verifying That Instances are Running
- Terminating Sessions On a Specific Cluster Instance
- Overview of Initialization Parameter Files in Oracle Real Application Clusters
- Initialization Parameter Use in Real Application Clusters
- Quiescing Oracle Real Application Clusters Databases
- Administering Multiple Cluster Interconnects on Linux and UNIX Platforms
- Customizing How Oracle Clusterware Manages Oracle RAC Databases
- Advanced Oracle Enterprise Manager Administration

> **See Also:** *Oracle Enterprise Manager Concepts* and the Enterprise Manager online help for more information about Enterprise Manager

Tools for Administering Oracle Real Application Clusters

The following sections introduce Oracle RAC administration using the three tools that you will most likely use to manage Oracle RAC databases and instances: Oracle Enterprise Manager, SQL*Plus, and the `SRVCTL` utility. In many cases, you use these tools the same way to manage Oracle RAC environments as you would use them manage single-instance Oracle databases:

- Administering Oracle Real Application Clusters with Enterprise Manager
- Administering Oracle Real Application Clusters with SQL*Plus
- Administering Oracle Real Application Clusters with SRVCTL

Administering Oracle Real Application Clusters with Enterprise Manager

Oracle Enterprise Manager provides a central point of control for the Oracle RAC environment, allowing you to perform administrative tasks simultaneously on multiple cluster databases. It has both the Database Control and Grid Control

graphical user interfaces (GUIs) for managing single-instance and Oracle RAC environments. Because there is one Enterprise Manager Agent on each node of an Oracle RAC database, for Database Control you can use any URL for that database to administer it with Enterprise Manager.

In Enterprise Manager, Oracle RAC-specific administrative tasks generally focus on two levels: tasks that affect an entire **cluster database** and tasks that affect specific instances. For example, you can use Enterprise Manager to start, stop, and monitor databases, cluster database instances, and their listeners, as well as to schedule jobs or set up alert thresholds for metrics. Or you can perform instance-specific commands such as setting parameters or creating resource plans. You can also use the Console to manage schemas, security, and cluster database storage features.

See Also:

- *Oracle Database 2 Day + Real Application Clusters Guide* for a task-oriented guide that explains how to use Enterprise Manager to perform routine Oracle RAC database administrative tasks
- "Advanced Oracle Enterprise Manager Administration" on page 3-20 for advanced administration tasks not covered in *Oracle Database 2 Day + Real Application Clusters Guide*

Administering Oracle Real Application Clusters with SQL*Plus

SQL*Plus commands operate on the current instance. The current instance can be either the local default instance on which you initiated your SQL*Plus session, or it can be a remote instance to which you connect with Oracle Net Services.

Because, by default, the SQL*Plus prompt does not identify the current instance, you should direct your commands to the correct instance. Starting a SQL*Plus session and connecting to the database without specifying an instance directs all SQL*Plus commands to the local instance. In this case, the default instance is also the current instance.

To connect to a different instance in SQL*Plus, issue a new `CONNECT` command and specify a remote instance net service name, as shown in the following example:

```
CONNECT user_name/password@net_service_name
```

Connecting as `SYSOPER` or `SYSDBA` enables you to perform privileged operations, such as instance startup and shutdown. Multiple SQL*Plus sessions can connect to the same instance at the same time. SQL*Plus automatically disconnects you from the first instance whenever you connect to another one.

Note: Use the `SYSASM` privilege instead of the `SYSDBA` privilege to connect to and administer an ASM instance. If you use the `SYSDBA` privilege to connect to an ASM instance, then Oracle Database writes warnings to the alert log files because commands that run using the `SYSDBA` privilege on an ASM instance are deprecated.

See the *Oracle Database Storage Administrator's Guide* for more information.

See Also:

- *SQL*Plus User's Guide and Reference*
- *Oracle Database Net Services Administrator's Guide* for the proper specification of `net_service_name`
- The *Oracle Database Administrator's Guide* for information about connecting to the database using `SYSDBA` or `SYSOPER` privileges

Changing the SQL*Plus Prompt

You may wish to change the SQL*Plus prompt so that it includes the name of the current instance. To do this you can issue a SQL*Plus command such as the following:

```
SET SQLPROMPT '_CONNECT_IDENTIFIER> '
```

This command replaces the "SQL" string in front of the greater than symbol (>) with the user variable `_CONNECT_IDENTIFIER` that will display the current instance name for the duration of your current session.

To change the prompt for all sessions automatically, add an entry similar to the following entry in your `glogin.sql` file, found in the SQL*Plus administrative directory:

```
SET SQLPROMPT '_CONNECT_IDENTIFIER> '
```

You may include any other required text or SQL*Plus user variable between the single quotes in the command.

How SQL*Plus Commands Affect Instances

Most SQL statements affect the current instance. You can use SQL*Plus to start and stop instances in the Oracle RAC database. You do not need to run SQL*Plus commands as `root` on Linux and UNIX systems or as `Administrator` on Windows systems. You need only the proper database account with the privileges that you normally use for a single-instance Oracle database. Some examples of how SQL*Plus commands affect instances are:

- `ALTER SYSTEM CHECKPOINT LOCAL` affects only the instance to which you are currently connected, rather than the default instance or all instances.
- `ALTER SYSTEM CHECKPOINT` or `ALTER SYSTEM CHECKPOINT GLOBAL` affects *all* instances in the cluster database.
- `ALTER SYSTEM SWITCH LOGFILE` affects only the current instance.
  - To force a global log switch, use the `ALTER SYSTEM ARCHIVE LOG CURRENT` statement.
  - The `INSTANCE` option of `ALTER SYSTEM ARCHIVE LOG` enables you to archive each online redo log file for a specific instance.

Table 3–1 describes how SQL*Plus commands affect instances.

Table 3–1 How SQL*Plus Commands Affect Instances

SQL*Plus Command	Associated Instance
`ARCHIVE LOG`	Always affects the current instance.
`CONNECT`	Affects the default instance if no instance is specified in the `CONNECT` command.

*Table 3–1 (Cont.) How SQL*Plus Commands Affect Instances*

SQL*Plus Command	Associated Instance
HOST	Affects the node running the SQL*Plus session, regardless of the location of the current and default instances.
RECOVER	Does not affect any particular instance, but rather the database.
SHOW INSTANCE	Displays information about the current instance, which can be different from the default local instance if you have redirected your commands to a remote instance.
SHOW PARAMETER and SHOW SGA	Displays parameter and SGA information from the current instance.
STARTUP and SHUTDOWN	Always affects the current instance. These are privileged SQL*Plus commands.

Administering Oracle Real Application Clusters with SRVCTL

The Server Control (SRVCTL) tool is a command-line interface that you can use to manage an Oracle RAC database from a single point. You can use SRVCTL to start and stop the database and instances, and to delete or move instances and services. You can also use SRVCTL to add services and manage configuration information. You use SRVCTL to start and stop a group of programs that includes virtual IP addresses, listeners, Oracle Notification Services, node-level applications.

The SRVCTL tool also manages configuration information that is used by several other Oracle tools. For example, Enterprise Manager uses the configuration information that SRVCTL generates to discover and monitor nodes in your cluster.

When you use SRVCTL to perform configuration operations on your cluster, SRVCTL stores configuration data in the Oracle Cluster Registry (OCR). SRVCTL performs other operations, such as starting and stopping instances, by calling SQL*Plus on each node.

Starting and Stopping Instances and Oracle Real Application Clusters Databases

You can start up and shut down instances with Enterprise Manager, SQL*Plus or SRVCTL as described in the following sections. Both Enterprise Manager and SRVCTL provide options to startup and shutdown all of the instances in an Oracle RAC database with a single step.

You can only perform certain operations when the database is in a NOMOUNT or MOUNT state. Performing other operations requires that the database be OPEN. In addition, some operations require that only one instance be in the required state, while other operations require that all of the instances be in an identical state.

The procedures in the following sections assume that you are using a server parameter file (SPFILE):

- Starting and Stopping with Enterprise Manager
- Starting Up and Shutting Down with SQL*Plus
- Starting Up and Shutting Down with SRVCTL

Before you can start an Oracle RAC instance, your clusterware and any required operating system-specific processes must be running. For more information about these processes, see your operating system documentation.

Note: After a cluster node restart, the node may not be fully responsive for some period of time. During this time, Oracle Database is attempting to restart the `OracleDBConsole`*`sid`* process and the `OracleCRService` resource. Eventually, all of the resource startup operations will complete and the computer will operate normally.

The procedure for shutting down Oracle RAC instances is identical to shutting down instances in single-instance Oracle, with the exceptions described here. See the *Oracle Database Administrator's Guide* for more information about shutting down Oracle databases.

- In Oracle RAC, shutting down one instance does not interfere with the operation of other running instances.
- To shut down an Oracle RAC database completely, shut down every instance that has the database open or mounted.
- After a `NORMAL` or `IMMEDIATE` shutdown, instance recovery is not required. Recovery is required, however, after you issue the `SHUTDOWN ABORT` command or after an instance terminates abnormally. The instance that is still running performs instance recovery for the instance that shut down. If no other instances are running, the next instance to open the database performs instance recovery for any instances needing it.
- `The SHUTDOWN TRANSACTIONAL` command with the `LOCAL` option is useful to shutdown an instance after all active transactions on the instance have either committed or rolled back. This is in addition to what this command does for `SHUTDOWN IMMEDIATE`. Transactions on other instances do not block this operation. If you omit the `LOCAL` option, then this operation waits until transactions on all other instances that started before the shutdown was issued either commit or rollback.

Starting and Stopping with Enterprise Manager

See the *Oracle Database 2 Day + Real Application Clusters Guide* for step-by-step instructions to start up or shut down a cluster database instance or a cluster database.

Starting Up and Shutting Down with SQL*Plus

If you want to start or stop just one instance and you are connected to your local node, you should first ensure that your current environment includes the SID for the local instance. Note that any subsequent commands in your session, whether inside or outside a SQL*Plus session, will be associated with that same SID.

To start or shutdown your local instance, initiate a SQL*Plus session and connect with the `SYSDBA` or `SYSOPER` privilege and then issue the required command. For example to start and mount an instance on your local node, run the following commands in your SQL*Plus session:

```
CONNECT / AS SYSDBA
STARTUP MOUNT
```

> **Note:** If you use ASM disk groups, use the SYSASM privilege instead of the SYSDBA privilege to connect to and administer the ASM instances. See the *Oracle Database Storage Administrator's Guide* for more information.

You can start more than one instance from a single SQL*Plus session on one node by way of Oracle Net Services. To achieve this, you must connect to each instance in turn by using a Net Services connection string, typically an instance-specific alias from your TNSNAMES.ORA file.

> **Note:** To ensure that you connect to the correct instance, you must use an alias in the connect string that is associated with just one instance. If you use an alias to a service or with multiple addresses, you may not be connected to your intended instance.

For example, you can use a SQL*Plus session on a local node to perform a transactional shutdown for two instances on remote nodes by connecting to each in turn using the instance's individual alias name. Assume the alias name for the first instance is db1 and that the alias for the second instance is db2. Connect to the first instance and shut it down as follows:

```
CONNECT /@db1 AS SYSDBA
SHUTDOWN TRANSACTIONAL
```

Then connect to and shutdown the second instance by entering the following from you SQL*Plus session:

```
CONNECT /@db2 AS SYSDBA
SHUTDOWN TRANSACTIONAL
```

It is not possible to start up or shut down more than one instance at a time in SQL*Plus, so you cannot start or stop all of the instances for a cluster database with a single SQL*Plus command. You may want to create a script that will connect to each instance in turn and start it up and shut it down. However, you must maintain this script manually if you add or drop instances.

> **See Also:** *SQL*Plus User's Guide and Reference*for information about other startup and shut down keywords, such as NOMOUNT, MOUNT, IMMEDIATE, and so on

Starting Up and Shutting Down with SRVCTL

Enter the following SRVCTL syntax from the command line, providing the required database name and instance name, or include more than one instance name to start more than one specific instance:

```
srvctl start instance -d db_name -i "inst_name_list" [-o start_options]
```

Note that this command will also start all enabled and non-running services that have the listed instances either as preferred or available instances.

To stop one or more instances, enter the following SRVCTL syntax from the command line:

```
srvctl stop instance -d name -i "inst_name_list" [-o stop_options]
```

This command also stops the services related to the terminated instances on the nodes where the instances were running. As an example, the following command provides its own connection information to shut down the two instances, `orcl3` and `orcl4`, using the `IMMEDIATE` option:

```
srvctl stop instance -d orcl -i "orcl3,orcl4" -o immediate
```

To start or stop your entire cluster database, that is, all of the instances and its enabled services, enter the following `SRVCTL` commands:

```
srvctl start database -d name [-o start_options]

srvctl stop database -d name [-o stop_options]
```

The following `SRVCTL` command, for example, mounts all of the non running instances of an Oracle RAC database using the default connection information:

```
srvctl start database -d orcl -o mount
```

> **See Also:** Appendix A, "Server Control Utility Reference" for information about `SRVCTL` options and information about other administrative tasks that you can perform with `SRVCTL`

Verifying That Instances are Running

To verify that instances are running, on any node from a SQL*Plus prompt enter:

```
CONNECT SYS/password as SYSDBA
SELECT * FROM V$ACTIVE_INSTANCES;
```

This query returns output similar to the following:

```
INST_NUMBER INST_NAME
-----------  ------------------
          1 db1-sun:db1
          2 db2-sun:db2
          3 db3-sun:db3
```

The output columns for this example are shown in Table 3–2.

Table 3–2 Descriptions of V$ACTIVE_INSTANCES Columns

Column	Description
`INST_NUMBER`	Identifies the instance number.
`INST_NAME`	Identifies the host name and instance name as *`host_name`*:*`instance_name`*.

Terminating Sessions On a Specific Cluster Instance

You can use the `ALTER SYSTEM KILL SESSION` statement to terminate a session on a specific instance. When a session is terminated, any active transactions of the session are rolled back, and resources held by the session (such as locks and memory areas) are immediately released and available to other sessions.

Using this statement enables you to maintain strict application service-level agreements in Oracle RAC environments. Often, the goal of a service-level agreement is to execute a transaction in a specified time limit. In an Oracle RAC environment, this

may require terminating a transaction on an instance and retrying the transaction on another instance within a specified time frame.

To terminate sessions, follow these steps:

- Query the value of the INST_ID column in the GV$SESSION dynamic performance view to identify which session to terminate
- Issue the ALTER SYSTEM KILL SESSION and specify the session index number (SID) and serial number of a session that you identified with the GV$SESSION dynamic performance view.

  ```
  KILL SESSION 'integer1, integer2[, @integer3]'
  ```

  - For *integer1*, specify the value of the SID column.
  - For *integer2*, specify the value of the SERIAL# column.
  - For the optional *integer3*, specify the ID of the instance where the session to be killed exists. You can find the instance ID by querying the GV$ tables.

  To use this statement, your instance must have the database open, and your session and the session to be terminated must be on the same instance unless you specify *integer3*.

If the session is performing some activity that must be completed, such as waiting for a reply from a remote database or rolling back a transaction, then Oracle Database waits for this activity to complete, marks the session as terminated, and then returns control to you. If the waiting lasts a minute, then Oracle Database marks the session to be terminated and returns control to you with a message that the session is marked to be terminated. The PMON background process then marks the session as terminated when the activity is complete.

The following examples provide a three scenarios in which a user identifies and terminates a specific session.

Example 1 Identify and terminate the session on an busy instance

In this example, assume that the executing session is SYSDBA on the instance INST_ID=1. The SYSDBA first queries the GV$SESSION view for the SCOTT user's session to identify the session to terminate, and then issues the ALTER SYSTEM KILL SESSION statement to terminate the session on the instance INST_ID=2. The ORA-00031 message is returned because some activity must be completed before the session can be terminated.

```
SQL> SELECT SID, SERIAL#, INST_ID FROM GV$SESSION WHERE USERNAME='SCOTT';

       SID    SERIAL#    INST_ID
---------- ---------- ----------
        80          4          2

SQL> ALTER SYSTEM KILL SESSION '80, 4, @2';
alter system kill session '80, 4, @2'
*
ERROR at line 1:
ORA-00031: session marked for kill
SQL>
```

Example 2 Identify and terminate the session on an idle instance

In this example, assume that the executing session is SYSDBA on the instance INST_ID=1. The session on instance INST_ID=2 is terminated immediately when Oracle Database executes the statement within 60 seconds.

```
SQL> SELECT SID, SERIAL#, INST_ID FROM GV$SESSION WHERE USERNAME='SCOTT';

       SID    SERIAL#    INST_ID
---------- ---------- ----------
        80          6          2

SQL> ALTER SYSTEM KILL SESSION '80, 6, @2';

System altered.

SQL>
```

Example 3 Using the IMMEDIATE parameter

The following example include the optional `IMMEDIATE` clause to immediately terminate the session without waiting for outstanding activity to complete.

```
SQL> SELECT SID, SERIAL#, INST_ID FROM GV$SESSION WHERE USERNAME='SCOTT';

       SID    SERIAL#    INST_ID
---------- ---------- ----------
        80          8          2

SQL> ALTER SYSTEM KILL SESSION '80, 8, @2' IMMEDIATE;

System altered.

SQL>
```

> **See Also:** *Oracle Database Administrator's Guide* for more information about terminating sessions

Overview of Initialization Parameter Files in Oracle Real Application Clusters

When you create the database, Oracle Database creates an SPFILE in the file location that you specify. This location can be an ASM disk group, a cluster file system, or a shared raw device. If you manually create your database, then Oracle recommends that you create an SPFILE from an initialization parameter file (PFILE).

> **Note:** Oracle RAC uses a traditional PFILE only if an SPFILE does not exist or if you specify `PFILE` in your `STARTUP` command. Oracle recommends that you use SPFILE file to simplify administration, to maintain parameter setting consistency, and to guarantee parameter setting persistence across database shutdown and startup events. In addition, you can configure RMAN to back up your SPFILE.

All instances in the cluster database use the same SPFILE at startup. Because the SPFILE is a binary file, do not directly edit the SPFILE with an editor. Instead, change SPFILE parameter settings using Enterprise Manager or `ALTER SYSTEM` SQL statements.

If you include the `FROM MEMORY` clause (for example, `CREATE PFILE FROM MEMORY` or `CREATE SPFILE FROM MEMORY`), the `CREATE` statement creates a PFILE or SPFILE using the current systemwide parameter settings. In an Oracle RAC environment, the created file contains the parameter settings from each instance. Because the `FROM MEMORY` clause requires all other instances to send their parameter

settings to the instance that is trying to create the parameter file, the total execution time depends on the number of instances, the number of parameter settings on each instance, and the amount of data for these settings.

Setting SPFILE Parameter Values for Oracle Real Application Clusters

You can alter SPFILE settings with Enterprise Manager or by using the `SET` clause of the `ALTER SYSTEM` statement.

> **Note:** Modifying the SPFILE using tools other than Enterprise Manager or SQL*Plus can corrupt the file and prevent database startup. To repair the file, you might need to create a PFILE and then regenerate the SPFILE.

The examples in this section appear in ASCII text although the SPFILE is a binary file. Assume that you start an instance with an SPFILE containing the following entries:

```
*.OPEN_CURSORS=500
prod1.OPEN_CURSORS=1000
```

> **Note:** The value before the dot in an SPFILE entry identifies the instance to which the particular parameter value belongs. When an asterisk precedes the dot, the value is applied to all instances that do not have a subsequent, individual value listed in the SPFILE.

For the instance with the Oracle system identifier (SID) `prod1`, the `OPEN_CURSORS` parameter is set to `1000` even though it has a database-wide setting of `500`. Parameter file entries that have the asterisk (*) wildcard character only affect the instances without an instance-specific entry. This gives you control over parameter settings for instance `prod1`. These two types of settings can appear in any order in the parameter file.

If another DBA runs the following statement, then Oracle updates the setting on all instances except the instance with SID `prod1`:

```
ALTER SYSTEM SET OPEN_CURSORS=1500 sid='*' SCOPE=MEMORY;
```

Then if you run the following statement on another instance, the instance with sid `prod1` also assumes the new setting of `2000`:

```
ALTER SYSTEM SET OPEN_CURSORS=2000 sid='*' SCOPE=MEMORY;
```

In the following example, the server parameter file contains these entries:

```
prod1.OPEN_CURSORS=1000
*.OPEN_CURSORS=500
```

Issue the following statement to make Oracle disregard the first entry from the server parameter file:

```
ALTER SYSTEM RESET SCOPE=SPFILE;
```

Issue the following statement to reset a parameter to its default value for instance `prod1` only:

```
ALTER SYSTEM RESET OPEN_CURSORS SCOPE=SPFILE SID='prod1';
```

Parameter File Search Order in Oracle Real Application Clusters

Oracle searches for your parameter file in a particular order depending on your platform.

On Linux and UNIX platforms, the search order is as follows:

1. `$ORACLE_HOME/dbs/spfilesid.ora`
2. `$ORACLE_HOME/dbs/spfile.ora`
3. `$ORACLE_HOME/dbs/initsid.ora`

On Windows platforms, the search order is as follows:

1. `%ORACLE_HOME%\database\spfilesid.ora`
2. `%ORACLE_HOME%\database\spfile.ora`
3. `%ORACLE_HOME%\database\initsid.ora`

Backing Up the Server Parameter File

Oracle recommends that you regularly back up the server parameter file for recovery purposes. Do this using Enterprise Manager (as described in the *Oracle Database 2 Day + Real Application Clusters Guide*) or use the `CREATE PFILE` statement. For example:

```
CREATE PFILE='?/dbs/initdbname.ora'
FROM SPFILE='/dev/vx/rdsk/oracle_dg/dbspfile'
```

You can use Recovery Manager (RMAN) to create backups of the server parameter file. You can also recover an SPFILE by starting an instance using a client-side initialization parameter file. Then re-create the server parameter file using the `CREATE SPFILE` statement. Note that if the parameter file that you use for this operation was for a single instance, then the parameter file will not contain instance-specific values, even those that must be unique in Oracle RAC instances. Therefore, ensure that your parameter file contains the appropriate settings as described earlier in this chapter.

To ensure that your SPFILE (and control files) are automatically backed up by RMAN during typical backup operations, use Enterprise Manager or the RMAN `CONTROLFILE AUTOBACKUP` statement to enable the RMAN autobackup feature

> **See Also:** *Oracle Database 2 Day + Real Application Clusters Guide* to perform backup jobs using Enterprise Manager, and the *Oracle Database SQL Language Reference* for more information about the `CREATE SPFILE` statement

Initialization Parameter Use in Real Application Clusters

By default, most parameters are set to a default value and this value is the same across all instances. However, many initialization parameters can also have different values on different instances as described in Table 3–3. Other parameters *must* either be unique or identical as described in the following sections.

- Parameters That Must Have Identical Settings on All Instances
- Parameters That Have Unique Settings on All Instances
- Parameters That Should Have Identical Settings on All Instances

Table 3–3 summarizes the initialization parameters used specifically for Oracle RAC databases. See *Oracle Database Reference* for additional information about these and other initialization parameters.

Table 3–3 Initialization Parameters Specific to Oracle Real Application Clusters

Parameter	Description
ASM_PREFERRED_READ_FAILURE_GROUPS	Specifies a set of disks to be the preferred disks from which to read mirror data copies. The values you set for this parameter are instance specific and need not be the same on all instances.
CLUSTER_DATABASE	Enables a database to be started in cluster mode. Set this parameter to TRUE.
CLUSTER_DATABASE_INSTANCES	Sets the number of instances in your Oracle RAC environment. A proper setting for this parameter can improve memory use. Set the CLUSTER_DATABASE_INSTANCES parameter to the *same* value on all instances. Otherwise, instance startup can fail. Normally, set this parameter to be equal to the number of instances in your Oracle RAC database. Alternatively, you can set this parameter to a value that is *greater than* the current number of instances if you are planning to add instances. **Note:** The value for this parameter determines the maximum number of instances that you can have in your Oracle RAC database environment. If you add instances, then you may need to reset the value for this parameter to accommodate the increased number of instances.
CLUSTER_INTERCONNECTS	Specifies an alternative cluster interconnect for the private network when there is more than one interconnect. The CLUSTER_INTERCONNECTS initialization parameter is used only in Linux and UNIX environments where UDP IPC is enabled. **Notes:** ■ Oracle recommends that all Oracle databases and Oracle Clusterware use the same interconnect network. If an Oracle database runs on a separate network from Oracle Clusterware, then the timeout on error is 900s. ■ Oracle does not recommend setting the CLUSTER_INTERCONNECTS parameter except in certain situations. See "Administering Multiple Cluster Interconnects on Linux and UNIX Platforms" on page 3-16 for more details.
DB_NAME	If you set a value for DB_NAME in instance-specific parameter files, the setting must be identical for all instances.
DISPATCHERS	Set the DISPATCHERS parameter to enable a shared server configuration, that is a server that is configured to enable many user processes to share very few server processes. With shared server configurations, many user processes connect to a dispatcher. The DISPATCHERS parameter may contain many attributes. Oracle recommends that you configure at least the PROTOCOL and LISTENER attributes. PROTOCOL specifies the network protocol for which the dispatcher process generates a listening end point. LISTENER specifies an alias name for the Oracle Net Services listeners. Set the alias to a name that is resolved through a naming method such as a tnsnames.ora file. The tnsnames.ora file contains net service names. Clients, nodes, and the Oracle Performance Manager node need this file. Enterprise Manager does not require tnsnames.ora entries on the client for Database Control or Grid Control. See *Oracle Database Net Services Administrator's Guide* for complete information about configuring the DISPATCHERS parameter and its attributes and for configuring the shared server.
GCS_SERVER_PROCESSES	This static parameter specifies the initial number of server processes for an Oracle RAC instance's Global Cache Service (GCS). The GCS processes manage the routing of interinstance traffic among Oracle RAC instances. The default number of GCS server processes is calculated based on system resources with a minimum setting of 2. For systems with one CPU, there is one GCS server process. For systems with two to eight CPUs, there are two GCS server processes. For systems with more than eight CPUs, the number of GCS server processes will be equal to the number of CPUs divided by 4, dropping any fractions. For example, if you have 10 CPUs, then 10 divided by 4 means that your system has 2 GCS processes. You can set this parameter to different values on different instances.
INSTANCE_NAME	Specifies the number of the redo log file groups to be used by an instance. See Redo Log File Storage in Oracle Real Application Clusters on page 2-2 for more information.

Table 3–3 (Cont.) Initialization Parameters Specific to Oracle Real Application Clusters

Parameter	Description
RESULT_CACHE_MAX_SIZE	In a clustered database, you can either set RESULT_CACHE_MAX_SIZE=0 on every instance to disable the result cache, or use a nonzero value on every instance to enable the result cache. To switch between enabled and disabled result cache requires that you restart every instance: ▪ **Enabling the result cache:** Set RESULT_CACHE_MAX_SIZE to a value greater than 0, or leave the parameter unset. You can size the cache differently on individual instances. ▪ **Disabling the result cache:** Set RESULT_CACHE_MAX_SIZE=0 on all instances to disable the result cache. If you set RESULT_CACHE_MAX_SIZE=0 upon startup of any one instance, then you must set the parameter to zero on all instance startups because disabling the result cache must done clusterwide. Disabling the result cache on some instances may lead to incorrect results. If you do not set the RESULT_CACHE_MAX_SIZE parameter, the parameter resolves to a default, nonzero value.
SERVICE_NAMES	When you use services, Oracle recommends that you do not set a value for the SERVICE_NAMES parameter but instead you should create cluster managed services through the Cluster Managed Services page in Enterprise Manager Database Control. This is because Oracle Clusterware controls the setting for this parameter for the services that you create and for the default database service. The service features described in Chapter 4, "Introduction to Automatic Workload Management" are not directly related to the features that Oracle provides when you set SERVICE_NAMES. In addition, setting a value for this parameter may override some of the benefits of using services. **Note:** Entries in the SERVICE_NAMES parameter may be used by client connections rather than the INSTANCE_NAME parameter value. The SERVICE_NAMES parameter may include one or more names and different instances may share one or more names with other instances. This enables a client to connect to either a specific instance or to any one of a set of instances, depending on the service name chosen in the connection string.
SESSIONS_PER_USER	Each instance maintains its own SESSIONS_PER_USER count. If SESSIONS_PER_USER is set to 1 for a user, the user can log on to the database more than once as long as each connection is from a different instance.
SPFILE	When you use an SPFILE, all Oracle RAC database instances must use the SPFILE and the file must be on shared storage.
THREAD	Specifies the number of the redo threads to be used by an instance. You can specify any available redo thread number as long as that thread number is enabled and is not used. If specified, this parameter must have unique values on all instances. The best practice is to use the INSTANCE_NAME parameter to specify redo log groups.

Parameters That Must Have Identical Settings on All Instances

Certain initialization parameters that are critical at database creation or that affect certain database operations must have the same value for every instance in an Oracle RAC database. Specify these parameter values in the SPFILE or in the individual PFILEs for each instance. The following list contains the parameters that must be identical on every instance:

```
ACTIVE_INSTANCE_COUNT
ARCHIVE_LAG_TARGET
COMPATIBLE
CLUSTER_DATABASE
CLUSTER_DATABASE_INSTANCE
CONTROL_FILES
DB_BLOCK_SIZE
DB_DOMAIN
```

DB_FILES
DB_NAME
DB_RECOVERY_FILE_DEST
DB_RECOVERY_FILE_DEST_SIZE
DB_UNIQUE_NAME
INSTANCE_TYPE (RDBMS or ASM)
PARALLEL_MAX_SERVERS
REMOTE_LOGIN_PASSWORD_FILE
UNDO_MANAGEMENT

The following parameters must be identical on every instance only if the parameter value is set to zero:

- DML_LOCKS
- RESULT_CACHE_MAX_SIZE

Parameters That Have Unique Settings on All Instances

If you use the ROLLBACK_SEGMENTS parameters, then Oracle recommends setting unique values for it by using the SID identifier in the SPFILE. However, you must set a unique value for INSTANCE_NUMBER for each instance and you cannot use a default value.

Oracle uses the INSTANCE_NUMBER parameter to distinguish among instances at startup. Oracle uses the INSTANCE_NAME parameter to assign redo log groups to specific instances. To simplify administration, use the same number for both the INSTANCE_NAME and INSTANCE_NUMBER parameters.

Specify the ORACLE_SID environment variable, which comprises the database name and the number of the INSTANCE_NAME assigned to the instance. When you specify UNDO_TABLESPACE with automatic undo management enabled, then set this parameter to a unique undo tablespace name for each instance.

Using the ASM_PREFERRED_READ_FAILURE_GROUPS initialization parameter, you can specify a list of preferred read failure group names. The disks in those failure groups become the preferred read disks. Thus, every node can read from its local disks. This results in higher efficiency and performance and reduced network traffic. The setting for this parameter is instance-specific, and the values need not be the same on all instances.

Parameters That Should Have Identical Settings on All Instances

Oracle recommends that you set the values for the parameters in Table 3–4 to the same value on all instances. Although you can have different settings for these parameters on different instances, setting each parameter to the same value on all instances simplifies administration.

Table 3–4 Parameters That Should Have Identical Settings on All Instances

Parameter	Description
ARCHIVE_LAG_TARGET	Different values for instances in your Oracle RAC database are likely to increase overhead because of additional automatic synchronization performed by the database processing. When using Streams with your Oracle RAC database, the value should be greater than zero.

Table 3–4 (Cont.) Parameters That Should Have Identical Settings on All Instances

Parameter	Description
LICENSE_MAX_USERS	Because this parameter determines a database-wide limit on the number of users defined in the database, it is useful to have the same value on all instances of your database so you can see the current value no matter which instance you are using. Setting different values may cause Oracle to generate additional warning messages during instance startup, or cause commands related to database user management to fail on some instances.
LOG_ARCHIVE_FORMAT	If you do not use the same value for all your instances, then you unnecessarily complicate media recovery. The recovering instance expects the required archive log file names to have the format defined by its own value of LOG_ARCHIVE_FORMAT, regardless of which instance created the archive log files. Databases that support Data Guard, either to send or receive archived redo log files, must use the same value of LOG_ARCHIVE_FORMAT for all instances.
SPFILE	If this parameter does not identify the same file to all instances, then each instance may behave differently and unpredictably in fail over, load-balancing, and during normal operations. Additionally, a change you make to the SPFILE with an ALTER SYSTEM SET or ALTER SYSTEM RESET command is saved only in the SPFILE used by the instance where you run the command. Your change will not be reflected in instances using different SPFILEs. If the SPFILE values are different in instances for which the values were set by the server, then you should restart the instances that are not using the default SPFILE.
TRACE_ENABLED	If you want diagnostic trace information to be always available for your Oracle RAC database, you must set TRACE_ENABLED to TRUE on all of your database instances. If you trace on only some of your instances, then diagnostic information might not be available when required should the only accessible instances be those with TRACE_ENABLED set to FALSE.
UNDO_RETENTION	By setting different values for UNDO_RETENTION in each instance, you are likely to reduce scalability and encounter unpredictable behavior following a fail over. Therefore, you should carefully consider whether you will accrue any benefits before you assign different values for this parameter to the instances in your Oracle RAC database.

Quiescing Oracle Real Application Clusters Databases

The procedure for quiescing Oracle RAC databases is identical to quiescing a single-instance database. You use the ALTER SYSTEM QUIESCE RESTRICTED statement from one instance. You cannot open the database from any instance while the database is in the process of being quiesced. Once all non-DBA sessions become inactive, the ALTER SYSTEM QUIESCE RESTRICTED statement finishes, and the database is considered as in a quiesced state. In an Oracle RAC environment, this statement affects all instances, not just the one from which the statement is issued.

To successfully issue the ALTER SYSTEM QUIESCE RESTRICTED statement in an Oracle RAC environment, you must have the Database Resource Manager feature activated, and it must have been activated since instance startup for all instances in the cluster database. It is through the facilities of the Database Resource Manager that non-DBA sessions are prevented from becoming active. Also, while this statement is in effect, any attempt to change the current resource plan will be queued until after the system is unquiesced.

These conditions apply to Oracle RAC:

- If you issued the `ALTER SYSTEM QUIESCE RESTRICTED` statement but Oracle Database has not finished processing it, you cannot open the database.
- You cannot open the database if it is already in a quiesced state.
- The `ALTER SYSTEM QUIESCE RESTRICTED` and `ALTER SYSTEM UNQUIESCE` statements affect all instances in an Oracle RAC environment, not just the instance that issues the command.

Note: You cannot use the quiesced state to take a cold backup. This is because Oracle background processes may still perform updates for Oracle internal purposes even while the database is in quiesced state. In addition, the file headers of online datafiles continue to look like they are being accessed. They do not look the same as if a clean shutdown were done. You can still take online backups while the database is in a quiesced state.

See Also: See the *Oracle Database Administrator's Guide* for details on the quiesce database feature and the *Oracle Database SQL Language Reference* for more information about the `ALTER SYSTEM QUIESCE RESTRICTED` syntax

Administering Multiple Cluster Interconnects on Linux and UNIX Platforms

In Oracle RAC environments that run on Linux and UNIX platforms where UDP IPC is enabled, you can use the `CLUSTER_INTERCONNECTS` initialization parameter to specify an alternative interconnect for the private network.

Oracle does not recommend setting the `CLUSTER_INTERCONNECTS` parameter, which overrides the default interconnect settings at the operating system level. Instead, the best practice is to use operating system bonding techniques (also referred to as NIC bonding). See your platform-specific Oracle Real Application Clusters installation guide for information about setting up NIC bonding at the operating system level.

This section contains the following topics:

- Recommendations for Setting the CLUSTER_INTERCONNECTS Parameter
- Usage Examples for the CLUSTER_INTERCONNECTS Parameter

Recommendations for Setting the CLUSTER_INTERCONNECTS Parameter

Oracle does not recommend setting the `CLUSTER_INTERCONNECTS` parameter.

Note: Oracle recommends that all databases and Oracle Clusterware use the same interconnect network.

Typically, you might need to set this the `CLUSTER_INTERCONNECTS` parameter only in the following situations:

- Due to operating system limitations, you cannot use Network Interface Card (NIC) bonding to provide increased bandwidth using multiple network interfaces.

- The cluster is running multiple databases and you need the interconnect traffic to be separated.
- You have a single IP address that is made highly available by the operating system, and it does not have a stable interface name (for example, the name can change when you restart).

Do not set the CLUSTER_INTERCONNECTS parameter for the following common configurations:

- If you have only one cluster interconnect.
- If the default cluster interconnect meets the bandwidth requirements of your Oracle RAC database, which is typically the case.

Consider the following important points when specifying the CLUSTER_INTERCONNECTS initialization parameter:

- The CLUSTER_INTERCONNECTS initialization parameter is useful only in Linux and UNIX environments where UDP IPC is enabled.
- Specify a different value for each instance of the Oracle RAC database when setting the CLUSTER_INTERCONNECTS initialization parameter in the parameter file.
- The IP addresses you specify for the different instances of the same database on different nodes must belong to network adapters that connect to the same interconnect network.
- The CLUSTER_INTERCONNECTS initialization parameter requires an IP address. It enables you to specify multiple IP addresses, separated by colons. Oracle RAC network traffic is distributed between the specified IP addresses.
- If you specify multiple IP addresses for this parameter, then list them in the same order for all instances of the same database. For example, if the parameter for instance 1 on node 1 lists the IP addresses of the alt0:, fta0:, and ics0: devices in that order, then the parameter for instance 2 on node 2 must list the IP addresses of the equivalent network adapters in the same order. See the examples in "Usage Examples for the CLUSTER_INTERCONNECTS Parameter" on page 3-17 for more information about setting multiple interconnects with this parameter.
- If an operating system error occurs while Oracle Database is writing to the interconnect that you specify with the CLUSTER_INTERCONNECTS parameter, then Oracle returns an error even if some other interfaces are available. This is because the communication protocols between Oracle Database and the interconnect can vary greatly depending on your platform. See your Oracle Database platform-specific documentation for more information.

> **See Also:** *Oracle Database Reference* for more information about the CLUSTER_INTERCONNECTS initialization parameter

Usage Examples for the CLUSTER_INTERCONNECTS Parameter

This section provides two examples for setting the CLUSTER_INTERCONNECTS parameter.

Example 1

Consider setting CLUSTER_INTERCONNECTS when a single cluster interconnect cannot meet your bandwidth requirements. You may need to set this parameter in data

warehouse environments with high interconnect bandwidth demands from one or more databases as described here.

For example, if you have two databases with high interconnect bandwidth requirements, then you can override the default interconnect provided by your operating system and nominate a different interconnect for each database using the following syntax in each server parameter file where `ip`*n* is an IP address in standard dot-decimal format, for example: `144.25.16.214`:

```
Database One: CLUSTER_INTERCONNECTS = ip1
Database Two: CLUSTER_INTERCONNECTS = ip2
```

If you have one database with high bandwidth demands, then you can nominate multiple interconnects using the following syntax:

```
CLUSTER_INTERCONNECTS = ip1:ip2:...:ipn
```

If you set multiple values for `CLUSTER_INTERCONNECTS` as in the preceding example, then Oracle Database uses all of the interconnects that you specify. This provides load balancing as long as all of the listed interconnects remain operational. You must use identical values, including the order in which the interconnects are listed, on all instances of your database when defining multiple interconnects with this parameter.

Example 2

To use the network interface whose IP address is `129.34.137.212` for all GCS, GES, and IPQ IPC traffic, set the `CLUSTER_INTERCONNECTS` parameter as follows:

```
CLUSTER_INTERCONNECTS=129.34.137.212
```

Use the `ifconfig` or `netstat` command to display the IP address of a device. This command provides a map between device names and IP addresses. For example, to determine the IP address of a device, run the following command as the `root` user:

```
# /usr/sbin/ifconfig -a
fta0: flags=c63<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST,SIMPLEX>
      inet 129.34.137.212 netmask fffffc00 broadcast 129.34.139.255 ipmtu 1500

lo0:  flags=100c89<UP,LOOPBACK,NOARP,MULTICAST,SIMPLEX,NOCHECKSUM>
      inet 127.0.0.1 netmask ff000000 ipmtu 4096

ics0:  flags=1100063<UP,BROADCAST,NOTRAILERS,RUNNING,NOCHECKSUM,CLUIF>
      inet 10.0.0.1 netmask ffffff00 broadcast 10.0.0.255 ipmtu 7000

sl0:  flags=10<POINTOPOINT>

tun0: flags=80<NOARP>
```

In the preceding example, the interface `fta0:` has an IP address of `129.34.137.212` and the interface `ics0:` has an IP address of `10.0.0.1`.

Customizing How Oracle Clusterware Manages Oracle RAC Databases

By default, Oracle Clusterware controls database restarts in Oracle RAC environments. In some cases, you may need to minimize the level of control that Oracle Clusterware has over your Oracle RAC database. You may need to do this, for example, during database upgrades or routine maintenance.

Note: When using third-party clusterware, Oracle recommends that you allow Oracle Database to manage the Oracle RAC instances. If you set the instance to manual and start it with third-party clusterware, do not use the third-party clusterware to monitor and restart database instances, Oracle Clusterware must do that.

To prevent Oracle Clusterware from restarting your Oracle RAC database when you restart your system, or to avoid restarting failed instances more than once, configure a policy to define the degree of control. There are two policies: automatic, which is the default, and manual. The manual policy minimizes the database instance protection level and overrides the automatic policy.

These policies enable you to configure your system so that either Oracle Clusterware automatically restarts your Oracle RAC database when you restart your system, or you manually restart your Oracle RAC database. You can also use this procedure to configure your system to prevent Oracle Clusterware from auto-restarting failed database instances more than once.

Use `SRVCTL` commands to display and change the Oracle Clusterware policies, as shown in the following examples:

Example 1 Display the Current Policy

For example, use the following command syntax to display the current policy where *`database_name`* is the name of the database for which you want to change policies:

```
srvctl config database -d database_name -a
```

Example 2 Change the Current Policy to Another Policy

Use the following `SRVCTL` command syntax to change the current policy to another policy where *`policy_name`* is the name of the new policy for the database (that was identified by *`database_name`* in Example1):

```
srvctl modify database d management_policy -y policy_name
```

This command syntax changes the resource profile values for each applicable resource and sets the Current Policy OCR key to the new value.

Example 3 Specify a Policy for a New Database

When you add a new database using the `SRVCTL` command, you can use the `-y` option to specify the management policy, as shown in the following example where *`database_name`* is the name of the database and *`management_policy`* is the name of the policy:

```
srvctl add database -d database_name -y management_policy
```

This command syntax places the new database under the control of Oracle Clusterware. If you do not provide a new management policy option, then Oracle uses the default value of `automatic`. After you change the policy, the OCR records the new value for the affected database.

See Also: Appendix A, "Server Control Utility Reference" for more information about `SRVCTL` commands

Advanced Oracle Enterprise Manager Administration

You can install, configure, and monitor an Oracle RAC database from a single location using either Oracle Enterprise Manager Database Control or Oracle Enterprise Manager Grid Control.

This section provides advanced administration tasks that are not covered in *Oracle Database 2 Day + Real Application Clusters Guide* or in "Overview of Monitoring and Tuning Oracle Real Application Clusters Databases" on page 11-1.

> **See Also:** *Oracle Database 2 Day + Real Application Clusters Guide* for a task-oriented guide that explains how to use Enterprise Manager to perform routine Oracle RAC database administrative tasks

This section includes the following topics:

- Using Enterprise Manager Grid Control to Discover Nodes and Instances
- Administering Jobs and Alerts in Oracle Real Application Clusters

Using Enterprise Manager Grid Control to Discover Nodes and Instances

Discovering Oracle RAC database and instance targets in Enterprise Manager enables monitoring and administration from the console:

- **Database Control** does not require discovery because DBCA performs any necessary configuration while creating the database.
- **Grid Control** enables you to use the Enterprise Manager console interface to discover Oracle RAC database and instance targets.

  If the Grid Control agents are installed on a cluster that already has an Oracle RAC database, Oracle RAC database targets are discovered at install time. You can use the console interface to discover targets if a database is created after agents are installed or if a database is not automatically discovered at agent install time.

To discover nodes and instances, use Enterprise Manager Grid Control as follows:

1. Log in to Enterprise Manager and click the **Targets** tab.
2. Click the **Database** tab to view all of the available targets. The column labeled **Types** shows the Oracle RAC databases using the entry "Cluster Database."
3. Add the database target by selecting the target name, then clicking **Add**. The Add Database Target: Specify Host page appears, which enables you to add databases, listeners, and Automatic Storage Management (ASM) as monitored targets.
4. Click the flashlight icon to display the available host names, select a host, then click **Continue**. The Add Database: Specify Source page appears.
5. Either request Enterprise Manager to discover only single-instance databases and listeners, or to discover all cluster databases, single-instance databases, and listeners on the cluster, then click **Continue**.

   Enterprise Manager performs discovery to locate and display the cluster database and its associated instances. The Targets Discovered on Cluster page appears. If this procedure did not discover your reconfigured cluster database and all of its instances, you can use this page to manually configure your cluster databases and single-instance databases.

Administering Jobs and Alerts in Oracle Real Application Clusters

The Cluster Database Home page shows all of the instances in the Oracle RAC database and provides an aggregate collection of several Oracle RAC-specific statistics that are collected by the **Automatic Workload Repository (AWR)** for server manageability.

You do not need to navigate to an instance-specific page to see these details. However, on the Cluster Database Home page, if an instance is down that should be operating, or if an instance has a high number of alerts, then you can drill down to the instance-specific page for each alert.

To perform specific administrative tasks as described in the remainder of this section, log in to the target Oracle RAC database, navigate to the Cluster Database Home page, and click the **Administration** tab.

Administering Jobs in Oracle Real Application Clusters

You can administer Enterprise Manager jobs at both the database and instance levels. For example, you can create a job at the cluster database level and the job will run on any active instance of the target Oracle RAC database. Or you can create a job at the instance level and the job will only run on the specific instance for which you created it. In the event of a failure, recurring jobs can run on a surviving instance.

Because you can create jobs at the instance level, cluster level, or cluster database level, jobs can run on any available host in the cluster database. This applies to scheduled jobs as well. Enterprise Manager also displays job activity in several categories, including, `Active`, `History`, and `Library`.

Use the Jobs tab to submit operating system scripts and SQL scripts and to examine scheduled jobs. For example, to create a backup job for a specific Oracle RAC database:

1. Click **Targets** and click the database for which you want to create the job.
2. Log in to the target database.
3. When Enterprise Manager displays the Database Home page, click **Maintenance**.
4. Complete the Enterprise Manage Job Wizard panels to create the job.

Administering Alerts in Oracle Real Application Clusters with Enterprise Manager

You can use Enterprise Manager to configure Oracle RAC environment alerts. You can also configure special Oracle RAC database tests, such as global cache converts, consistent read requests, and so on.

Enterprise Manager distinguishes between database- and instance-level alerts in Oracle RAC environments. Alert thresholds for instance level alerts, such as archive log alerts, can be set at the instance target level. This enables you to receive alerts for the specific instance if performance exceeds your threshold. You can also configure alerts at the database level, such as setting alerts for tablespaces. This enables you to avoid receiving duplicate alerts at each instance.

> **See Also:** Oracle Technology Network for an example of configuring alerts in Oracle RAC and the *Oracle Database PL/SQL Packages and Types Reference* for information about using packages to configure thresholds

Performing Scheduled Maintenance Using Defined Blackouts in Enterprise Manager

You can define blackouts for all managed targets of an Oracle RAC database to prevent alerts from occurring while performing maintenance. You can define blackouts for an entire cluster database or for specific cluster database instances.

4

Introduction to Automatic Workload Management

This chapter describes how to manage workloads in Oracle Real Application Clusters (Oracle RAC) to provide high availability and scalability for your applications. This chapter contains the following topics:

- Overview of Automatic Workload Management
- Automatic Workload Repository
- Service Deployment Options
- Fast Application Notification
- Load Balancing Advisory
- Oracle Clients That Are Integrated with Fast Application Notification
- Enabling Event Notification for Connection Failures in Oracle Real Application Clusters
- Services and Distributed Transaction Processing in Oracle Real Application Clusters
- Enabling Distributed Transaction Processing for Services
- Administering Services
- Administering Services with Enterprise Manager, PL/SQL, and SRVCTL
- Measuring Performance by Service Using the Automatic Workload Repository
- Service Thresholds and Alerts

Overview of Automatic Workload Management

Automatic workload management enables you to manage workload distributions to provide optimal performance for users and applications. Automatic workload management comprises the following:

- **Services**—Oracle Database provides a powerful automatic workload management facility, called services, to enable the enterprise grid vision. Services are entities that you can define in Oracle RAC databases that enable you to group database workloads and route work to the optimal instances that are assigned to offer the service.
- **Connection Load Balancing**—A feature of Oracle Net Services that balances incoming connections across all of the instances that provide the requested database service.

- **High Availability Framework**—A Oracle RAC component that enables the Oracle Database to maintain components in a running state at all times.
- **Fast Application Notification (FAN)**—The notification mechanism that Oracle RAC uses to quickly alert applications about cluster state changes and workload service level changes, such as `UP` and `DOWN` events for instances, services, or nodes.
- **Load Balancing Advisory**—Provides information to applications about the current service levels that the database and its instances are providing. The load balancing advisory makes recommendations to applications about where to direct application requests to obtain the best service based on the policy that you have defined for that service.
- **Automatic Workload Repository (AWR)**—Tracks service level statistics as metrics. Server generated alerts can be placed on these metrics when they exceed or fail to meet certain thresholds. You can then respond, for example, by changing the priority of a job, stopping overloaded processes, or by modifying a service level requirement. This enables you to maintain continued service availability despite service level changes. You can configure the service level for one service to have priorities relative to other services, and you can also configure:
  - The measurement of service quality
  - Event notification and alert mechanisms to monitor service quality changes
  - Recovery scenarios for responses to service quality changes

  The Automatic Workload Repository ensures that the Oracle Clusterware workload management framework and resource manager have persistent and global representations of performance data. This information helps Oracle schedule job classes by service and to assign priorities to consumer groups. If necessary, you can rebalance workloads manually with the `DBMS_SERVICE.DISCONNECT_SESSION` PL/SQL procedure. You can also use this procedure to disconnect a series of sessions and leave the service running.

  See Also: *Oracle Database Performance Tuning Guide* for details about the Automatic Workload Repository and *Oracle Database PL/SQL Packages and Types Reference* for details about Oracle packages
- **Fast Connection Failover**—This is the ability of Oracle Clients to provide rapid failover of connections by subscribing to FAN events.
- **Runtime Connection Load Balancing**—This is the ability of Oracle Clients to provide intelligent allocations of connections in the connection pool based on the current service level provided by the database instances when applications request a connection to complete some work.

When a user or application connects to a database, Oracle recommends that you specify a service in the connect data portion of the connect string. Oracle Database automatically creates one database service when the database is created. For many installations, this may be all you need. To enable more flexibility in the management of the workload using the database, Oracle Database enables you to create multiple services and specify which instances offer the services. Continue reading this chapter to understand the added features that you can use with services if you are interested in greater workload management flexibility.

You can deploy Oracle RAC and single-instance Oracle database environments to use automatic workload management features in many different ways. Depending on the number of nodes and your environment's complexity and objectives, your choices for

optimal automatic workload management and high-availability configuration depend on several considerations that this chapter describes. The following section describes the various service deployment options.

Automatic Workload Repository

The **Automatic Workload Repository (AWR)** tracks service level statistics as metrics. Server generated alerts can be placed on these metrics when they exceed or fail to meet certain thresholds. You can then respond, for example, by changing the priority of a job, stopping overloaded processes, or by modifying a service level requirement. This enables you to maintain continued service availability despite service level changes. You can configure the service level for one service to have priorities relative to other services, and you can also configure:

Service Deployment Options

This section describes the following service deployment topics:

- Using Oracle Services
- Default Service Connections
- Connection Load Balancing

Using Oracle Services

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

Oracle recommends that all users who share a service have the same service level requirements. You can define specific characteristics for services and each service can be a separate unit of work. There are many options that you can take advantage of when using services. Although you do not have to implement these options, using them helps optimize application performance.

When you define a service, you define which instances normally support that service. These are known as the `PREFERRED` instances. You can also define other instances to support a service if the service's preferred instance fails. These are known as `AVAILABLE` instances.

When you specify `PREFERRED` instances, you are specifying the number of instances on which a service will normally run. Oracle Clusterware attempts to ensure that the service always runs on the number of instances for which you have configured the service. Afterwards, due to either instance failure or planned service relocations, a service may be running on an `AVAILABLE` instance. You cannot control which `AVAILABLE` instance to which Oracle Clusterware relocates the services if there is more than one in the list. When a service moves to an `AVAILABLE` instance, Oracle does not move the service back to the `PREFERRED` instance when the `PREFERRED` instance restarts because:

- The service is already running on the desired number of instances
- Maintaining the service on the current instance provides a higher level of service availability
- Not moving the service back to the initial `PREFERRED` instance prevents a second outage

You can, however, easily automate fail back by using FAN callouts.

Resource profiles are automatically created when you define a service. A resource profile describes how Oracle Clusterware should manage the service and which instance the service should failover to if the preferred instance stops. Resource profiles also define service dependencies for the instance and the database. Due to these dependencies, if you stop a database, then the instances and services are automatically stopped in the correct order.

Services are integrated with Resource Manager, which enables you to restrict the resources that are used by the users who connect with a service in an instance. The Resource Manager enables you to map a consumer group to a service so that users who connect with the service are members of the specified consumer group. Also, the Automatic Workload Repository (AWR) enables you to monitor performance by service.

Oracle Net Services provides connection load balancing to enable you to spread user connections across all of the instances that are supporting a service. For each service, you can define the method that you want the listener to use for load balancing by setting the connection load balancing goal, `CLB_GOAL`. You can also specify a single transparent application failover (TAF) policy for all users of a service by defining the `FAILOVER_METHOD`, `FAILOVER_TYPE`, and so on. (See the *Oracle Database Net Services Administrator's Guide* for more information about configuring TAF.)

Oracle RAC uses FAN to notify applications about configuration changes and the current service level that is provided by each instance where the service is enabled. FAN has two methods for publishing events to clients, the Oracle Notification Service (ONS) that is used by Java Database Connectivity (JDBC) clients including the Oracle Application Server, and Oracle Streams, and Advanced Queueing that is used by Oracle Call Interface (OCI) and Oracle Data Provider for .NET (ODP.NET) clients. When using Advanced Queueing, you must enable the service to use the queue by setting `AQ_HA_NOTIFICATIONS` to true.

With runtime connection load balancing, applications can use load balancing advisory events to provide better service to users. The Oracle JDBC and `ODP.NET` clients are automatically integrated to take advantage of load balancing advisory events. The load balancing advisory informs the client about the current service level that an instance is providing for a service. The load balancing advisory also recommends how much of the workload should be sent to that instance. In addition, Oracle Net Services provides connection load balancing to enable you to spread user connections across all of the instances that support a service. To enable the load balancing advisory, set the `GOAL` parameter on the service.

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

> **See Also:** "Services and Distributed Transaction Processing in Oracle Real Application Clusters" on page 4-22 for more information about distributed transaction processing in Oracle RAC

Default Service Connections

A special Oracle database service is created by default for your Oracle RAC database. This default service is always available on all instances in an Oracle RAC environment, unless an instance is in restricted mode. You cannot alter this service or its properties. The database also supports the following two internal services:

- SYS$BACKGROUND is used by the background processes only
- SYS$USERS is the default service for user sessions that are not associated with any application service

Both of these internal services support all of the automatic workload management features. You cannot stop or disable either of these internal services.

> **Note:** You can explicitly manage only the services that you create. If a feature of the database creates an internal service, you cannot manage it using the information in this chapter.

Connection Load Balancing

Oracle Net Services provides the ability to balance client connections across the instances in an Oracle RAC configuration. There are two types of load balancing that you can implement: client-side and server-side load balancing. Client-side load balancing balances the connection requests across the listeners. With server-side load balancing, the listener directs a connection request to the best instance currently providing the service by using the load balancing advisory. In an Oracle RAC database, client connections should use both types of connection load balancing.

FAN, Fast Connection Failover, and the load balancing advisory depend on an accurate connection load balancing configuration that includes setting the connection load balancing goal for the service. You can use a goal of either LONG or SHORT for connection load balancing. These goals have the following characteristics:

- **LONG**—Use the LONG connection load balancing method for applications that have long-lived connections. This is typical for connection pools and SQL*Forms sessions. LONG is the default connection load balancing goal. The following is an example of modifying a service, POSTMAN, with the PL/SQL DBMS_SERVICE package and the CLB_GOAL_LONG package constant to define the connection load balancing goal for long-lived sessions:

```
EXECUTE DBMS_SERVICE.MODIFY_SERVICE (service_name => 'POSTMAN'
        , clb_goal => DBMS_SERVICE.CLB_GOAL_LONG);
```

- **SHORT**—Use the SHORT connection load balancing method for applications that have short-lived connections. The following example modifies the service known as ORDER, using the DBMS_SERVICE.CLB_GOAL_SHORT package constant to set the goal to SHORT:

```
EXECUTE DBMS_SERVICE.MODIFY_SERVICE (service_name => 'ORDER'
        , CLB_GOAL => DBMS_SERVICE.CLB_GOAL_SHORT);
```

When you create an Oracle RAC database with the DBCA, it automatically:

- Configures and enables server-side load balancing
- Sets the local and remote listener parameters (**Note:** If you do not use DBCA, you should set the LOCAL_LISTENER and REMOTE_LISTENER database parameters manually, especially if you do not use port 1521.)
- Creates a sample client-side load balancing connection definition in the tnsnames.ora file on the server

When Oracle Net Services establishes a connection to an instance, the connection remains open until the client closes the connection, the instance is shutdown, or a failure occurs. If you configure TAF for the connection, then Oracle Database moves the session to a surviving instance when an outage occurs.

TAF can restart a query after failover has completed but for other types of transactions, such as INSERT, UPDATE, or DELETE, the application must rollback the failed transaction and resubmit the transaction. You must reexecute any session customizations, in other words, ALTER SESSION statements, after failover has occurred. However, with TAF, a connection is not moved during normal processing, even if the workload changes over time.

Services simplify the deployment of TAF. You can define a TAF policy for a service and all connections using this service will automatically have TAF enabled. This does not require any client-side changes. The TAF setting on a service overrides any TAF setting in the client connection definition. To define a TAF policy for a service, use the DBMS_SERVICE PL/SQL procedure as in the following example:

```
EXECUTE DBMS_SERVICE.MODIFY_SERVICE (service_name => 'gl.us.oracle.com'
, aq_ha_notifications => TRUE
, failover_method => DBMS_SERVICE.FAILOVER_METHOD_BASIC
, failover_type => DBMS_SERVICE.FAILOVER_TYPE_SELECT
, failover_retries => 180
, failover_delay => 5
, clb_goal => DBMS_SERVICE.CLB_GOAL_LONG);
```

> **See Also:** *Oracle Database Net Services Administrator's Guide* for more information about configuring TAF

Client-side load balancing is defined in your client connection definition by setting the parameter LOAD_BALANCE=ON (the default is ON for description lists). When you set this parameter to ON, Oracle randomly selects an address in the address list, and connects to that node's listener. This balances client connections across the available listeners in the cluster. When the listener receives the connection request, the listener connects the user to an instance that the listener knows provides the requested service. To see what services a listener supports, run the lsnrctl services command.

In addition to client-side load balancing, Oracle Net Services include connection failover. If an error is returned from the chosen address in the list, Oracle Net Services will try the next address in the list until it is either successful or it has exhausted all addresses in its list. To increase availability, you can specify a timeout within which Oracle Net will wait for a response from the listener.

You can avoid delays by setting the oracle.net.ns.SQLnetDef.TCP_CONNTIMEOUT_STR property, as follows:

```
Properties prop = new Properties ();
prop.put (oracle.net.ns.SQLnetDef.TCP_CONNTIMEOUT_STR,
"" + (1 * 1000)); // 1 second
dbPools[ poolIndex ].setConnectionProperties ( prop );
```

The parameter value is specified in milliseconds. Therefore, it is possible to reduce the timeout to 500Ms if the application retries connecting.

For OCI clients, create a local sqlnet.ora file on the client side. Configure the connection timeout in this file by adding the following line:

```
sqlnet.outbound_connect_timeout = 1
```

The granularity of the timeout value for the OCI client is in seconds.

> **Note:** Do *not* configure the connection timeout in the sqlnet.ora file on the server.

See Also: *Oracle Database Net Services Administrator's Guide* for detailed information about both types of load balancing

Fast Application Notification

This section provides a detailed description of FAN under the following topics:

- Overview of Fast Application Notification
- Application High Availability with Services and FAN
- Managing Unplanned Outages
- Managing Planned Outages
- Fast Application Notification High Availability Events
- Using Fast Application Notification Callouts

See Also: "Oracle Clients That Are Integrated with Fast Application Notification" on page 4-12 for more information about specific client environments that you can use with FAN

Overview of Fast Application Notification

FAN is a notification mechanism that Oracle RAC uses to notify other processes about configuration and service level information that includes service status changes, such as UP or DOWN events. Applications can respond to FAN events and take immediate action. FAN `UP` and `DOWN` events can apply to instances, services, and nodes.

For cluster configuration changes, the Oracle RAC high availability framework publishes a FAN event immediately when a state change occurs in the cluster. Instead of waiting for the application to poll the database and detect a problem, applications can receive FAN events and react immediately.

FAN also publishes load balancing advisory events. Applications can take advantage of the load balancing advisory FAN events to direct work requests to the instance in the cluster that is currently providing the best service quality. You can take advantage of FAN events in the following three ways:

1. Your application can use FAN without programmatic changes if you use an integrated Oracle client. The integrated clients for FAN events include Oracle Database JDBC, Oracle Database ODP.NET, and Oracle Database OCI. This includes applications that use TAF. The integrated Oracle clients must be Oracle Database 10*g* Release 2 or later to take advantage of the load balancing advisory FAN events. (See the *Oracle Database Net Services Administrator's Guide* for more information about configuring TAF.)
2. Applications can use FAN programmatically by using the ONS Application Programming Interface (API) to subscribe to FAN events and to execute event handling actions upon the receipt of an event.
3. You can implement FAN with server-side callouts on your database tier.

For `DOWN` events, the disruption to the application can be minimized because sessions to the failed instance or node can be terminated. Incomplete transactions can be terminated and the application user is immediately notified. Application users who request connections are directed to available instances only. For `UP` events, when services and instances are started, new connections can be created so that the application can immediately take advantage of the extra resources. Through server-side callouts, you can also use FAN to:

- Log status information
- Page DBAs or to open support tickets when resources fail to start
- Automatically start dependent external applications that need to be co-located with a service
- Change resource plans or to shut down services when the number of available instances decreases, for example, if nodes fail
- Automate the fail back of a service to `PREFERRED` instances if needed

FAN events are published using ONS and an Oracle Streams Advanced Queuing. The publication mechanisms are automatically configured as part of your Oracle RAC installation.

The Connection Manager (CMAN) and Oracle Net Services listeners are integrated with FAN events. This enables the listener and CMAN to immediately de-register services provided by the failed instance and to avoid erroneously sending connection requests to failed instances.

If you use the service goal `CLB_GOAL_SHORT`, then the listener uses the load balancing advisory when the listener balances the connection loads. When load balancing advisory is enabled, the metrics used for the listener are finer grained.

Application High Availability with Services and FAN

Oracle Database focuses on maintaining service availability. In Oracle RAC, Oracle services are designed to be continuously available with loads shared across one or more instances. The Oracle RAC high availability framework maintains service availability by using Oracle Clusterware and resource profiles.

The Oracle RAC high availability framework monitors the database and its services and sends event notifications using FAN. Oracle Clusterware recovers and balances services according to business rules.

Managing Unplanned Outages

You can assign services in Oracle RAC to one or more instances. If Oracle RAC detects an outage, then Oracle Clusterware isolates the failed component and recovers the dependent components. For services, if the failed component is an instance, then Oracle Clusterware relocates the service to an available instance in the cluster. FAN events can occur at various levels within the Oracle Database architecture and are published through ONS and Advanced Queuing. You can also program notification using FAN callouts.

> **Note:** Oracle Database does not run Oracle RAC callouts with guaranteed ordering. Callouts are run asynchronously and they are subject to scheduling variabilities.

Notification occurs from a surviving node when the failed node is out of service. The location and number of instances in an Oracle RAC environment that provide a service are transparent to applications. Restart and recovery are automatic, including the restarting of the subsystems, such as the listener and the Automatic Storage Management (ASM) processes, not just the database. You can use FAN callouts to report faults to your fault management system and to initiate repair jobs.

Managing Planned Outages

For repairs, upgrades, and changes that require you to isolate one or more instances, Oracle RAC provides interfaces that relocate, disable, and enable services to minimize service disruption to application users. Once you complete the operation, you can return the service to normal operation.

Due to dependencies, if you manually shutdown your database, then all of your services automatically stop. If you then manually restart the database, then you must also restart the database's services. Use FAN callouts to automate starting the services when the database starts.

Fast Application Notification High Availability Events

Table 4–1 describes the FAN event record parameters and the event types, followed by name-value pairs for the event properties. The event type is always the first entry and the timestamp is always the last entry as in the following example:

```
FAN event type: service_member
Properties: version=1.0 service=ERP database=FINPROD instance=FINPROD3 host=node3
status=up
```

Table 4–1 Event Record Parameters and Descriptions

Parameter	Description
VERSION	Version of the event record. Used to identify release changes.
EVENT TYPE	SERVICE, SERVICE_MEMBER, DATABASE, INSTANCE, NODE, ASM, SRV_PRECONNECT. Note that Database and Instance types provide the database service, such as DB_UNIQUE_NAME.DB_DOMAIN.
DATABASE UNIQUE NAME	The unique database supporting the service; matches the initialization parameter value for DB_UNIQUE_NAME, which defaults to the value of the initialization parameter DB_NAME.
INSTANCE	The name of the instance that supports the service; matches the ORACLE_SID value.
NODE NAME	The name of the node that supports the service or the node that has stopped; matches the node name known to Cluster Synchronization Services (CSS).
SERVICE	The service name; matches the service in DBA_SERVICES.
STATUS	Values are UP, DOWN, NOT_RESTARTING, PRECONN_UP, PRECONN_DOWN, and UNKNOWN.
REASON	Failure, Dependency, User, Autostart, Restart.
CARDINALITY	The number of service members that are currently active; included in all UP events.
INCARNATION	For node DOWN events; the new cluster incarnation.
TIMESTAMP	The local time zone to use when ordering notification events.

A FAN record matches the database signature of each session as shown in Table 4–2.

Table 4–2 FAN Parameters and Matching Database Signatures

FAN Parameter	Matching Oracle Database Signature
SERVICE	sys_context('userenv', 'service_name')
DATABASE UNIQUE NAME	sys_context('userenv', 'db_unique_name')
INSTANCE	sys_context('userenv', 'instance_name')

Table 4–2 (Cont.) FAN Parameters and Matching Database Signatures

FAN Parameter	Matching Oracle Database Signature
CLUSTER NODE NAME	sys_context('userenv', 'server_host')

Using Fast Application Notification Callouts

FAN callouts are server-side executables that Oracle RAC executes immediately when high availability events occur. You can use FAN callouts to automate the following activities when events occur in a cluster configuration, such as:

- Opening fault tracking tickets
- Sending messages to pagers
- Sending e-mail
- Starting and stopping server-side applications
- Maintaining an uptime log by logging each event as it occurs
- Relocating low-priority services when high priority services come online

To use FAN callouts, place an executable in the directory *CRS_home*`/racg/usrco` on every node that runs Oracle Clusterware. If you are using scripts, then set the shell as the first line of the executable. The following is an example file for the *CRS_home*`/racg/usrco/callout.sh` callout:

```
#! /bin/ksh
FAN_LOGFILE= [your path name]/admin/log/`hostname`_uptime.log
echo $* "reported="`date` >> $FAN_LOGFILE &
```

The following output is from the previous example:

```
NODE VERSION=1.0 host=sun880-2 incarn=23 status=nodedown reason=
timestamp=08-Oct-2004 04:02:14 reported=Fri Oct 8 04:02:14 PDT 2004
```

Example callout scripts are available in the Oracle Real Application Clusters Sample Code section on Oracle Technology Network at
http://www.oracle.com/technology/sample_code/products/rac/

> **See Also:** Table 4–1, " Event Record Parameters and Descriptions" for information about the callout and event details

A FAN record matches the database signature of each session, as shown in Table 4–2, " FAN Parameters and Matching Database Signatures". The signature information is also available using `OCI_ATTRIBUTES`. These attributes are available in the OCI Connection Handle. Use this information to take actions on sessions that match the FAN event data.

Load Balancing Advisory

This section describes the load balancing advisory under the following topics:

- Overview of the Load Balancing Advisory
- Configuring Your Environment to Use the Load Balancing Advisory
- Load Balancing Advisory FAN Events

Overview of the Load Balancing Advisory

Load balancing distributes work across all of the available Oracle RAC database instances. Oracle recommends that applications use persistent connections that span the instances that offer a particular service. Connections are created infrequently and exist for a long duration. Work comes into the system with high frequency, borrows these connections, and exists for a relatively short duration. The load balancing advisory provides advice about how to direct incoming work to the instances that provide the optimal quality of service for that work. This minimizes the need to relocate the work later.

By using the THROUGHPUT or SERVICE_TIME goals, feedback is built in to the system. Work is routed to provide the best service times globally, and routing responds gracefully to changing system conditions. In a steady state, the system approaches equilibrium with improved throughput across all of the Oracle RAC instances.

Standard architectures that can use the load balancing advisory include connection load balancing, transaction processing monitors, application servers, connection concentrators, hardware and software load balancers, job schedulers, batch schedulers, and message queuing systems. All of these applications can allocate work.

The load balancing advisory is deployed with key Oracle clients, such as a listener, the JDBC Implicit Connection Cache, and the ODP.NET Connection Pool. The load balancing advisory is also open for third-party subscription by way of ONS.

Configuring Your Environment to Use the Load Balancing Advisory

You can configure your environment to use the load balancing advisory by defining service-level goals for each service for which you want to enable load balancing. This enables the load balancing advisory for that service and FAN load balancing events are published. There are two types of service-level goals for runtime:

- SERVICE TIME—Attempts to direct work requests to instances according to response time. Load balancing advisory data is based on elapsed time for work done in the service plus available bandwidth to the service. An example for the use of SERVICE TIME is for workloads such as internet shopping where the rate of demand changes:

```
EXECUTE DBMS_SERVICE.MODIFY_SERVICE (service_name => 'OE' -
, goal => DBMS_SERVICE.GOAL_SERVICE_TIME -
, clb_goal => DBMS_SERVICE.CLB_GOAL_SHORT);
```

- THROUGHPUT—Attempts to direct work requests according to throughput. The load balancing advisory is based on the rate that work is completed in the service plus available bandwidth to the service. An example for the use of THROUGHPUT is for workloads such as batch processes, where the next job starts when the last job completes:

```
EXECUTE DBMS_SERVICE.MODIFY_SERVICE (service_name => 'sjob' -
        , goal => DBMS_SERVICE.GOAL_THROUGHPUT -
       , clb_goal => DBMS_SERVICE.CLB_GOAL_LONG);
```

  Setting the goal to NONE disables load balancing for the service. You can see the goal settings for a service in the data dictionary and in the DBA_SERVICES, V$SERVICES, and V$ACTIVE_SERVICES views.

  See Also: "Administering Services" on page 4-24 for more information about administering services and adding goals to services

Load Balancing Advisory FAN Events

The load balancing advisory FAN events provide metrics for load balancing algorithms. The easiest way to take advantage of these events is to use the runtime connection load balancing feature of an Oracle integrated client such as JDBC, ODP.NET, or OCI. Client applications can subscribe to these events directly by way of the ONS API. Table 4–3 describes the load balancing advisory FAN event parameters.

Table 4–3 Load Balancing Advisory FAN Events

Parameter	Description
VERSION	Version of the event record. Used to identify release changes.
EVENT TYPE	SERVICE, SERVICE_MEMBER, DATABASE, INSTANCE, NODE, ASM, SRV_PRECONNECT. Note that Database and Instance types provide the database service, such as DB_UNIQUE_NAME.DB_DOMAIN.
SERVICE	The service name; matches the service in DBA_SERVICES.
DATABASE UNIQUE NAME	The unique database supporting the service; matches the initialization parameter value for DB_UNIQUE_NAME, which defaults to the value of the initialization parameter DB_NAME.
INSTANCE	The name of the instance that supports the service; matches the ORACLE_SID value.
PERCENT	The percentage of work requests to send to this database instance.
FLAG	Indication of the service quality relative to the service goal. Valid values are GOOD, VIOLATING, NO DATA, and BLOCKED.
TIMESTAMP	The local time zone to use when ordering notification events.

Use the following example to monitor load balancing advisory events:

```
SET PAGES 60 COLSEP '|' LINES 132 NUM 8 VERIFY OFF FEEDBACK OFF
COLUMN user_data HEADING "AQ Service Metrics" FORMAT A60 WRAP
BREAK ON service_name SKIP 1
SELECT
 TO_CHAR(enq_time, 'HH:MI:SS') Enq_time, user_data
 FROM sys.sys$service_metrics_tab
 ORDER BY 1 ;
```

Oracle Clients That Are Integrated with Fast Application Notification

Oracle has integrated FAN with many of the common client application environments that are used to connect to Oracle RAC databases. Therefore, the easiest way to use FAN is to use an integrated Oracle Client.

You can use the OCI Session Pools, CMAN session pools, and JDBC and ODP.NET connection pools. OCI applications with TAF enabled should use FAN high availability events for fast failover. The overall goal is to enable applications to consistently obtain connections to available instances that provide the best service. Due to the integration with FAN, Oracle integrated clients are more aware of the current status of an Oracle RAC cluster. This prevents client connections from waiting or trying to connect to an instance that is no longer available.

When instances start, Oracle RAC uses FAN to notify the connection pool so that the connection pool can create connections to the recently started instance and take advantage of the additional resources that this instance provides. The use of connection pools and FAN requires that you have properly configured database connection load balancing across all of the instances that provide the service(s) that the connection pool is using. Oracle recommends that you configure both client-side and

server-side load balancing with Oracle Net Services, which is the default when you use DBCA to create your database. Oracle connection pools that are integrated with FAN can:

- Balance connections across all of the Oracle RAC instances when a service starts; this is preferable to directing the sessions that are defined for the connection pool to the first Oracle RAC instance that supports the service
- Remove terminated connections immediately when a service is declared `DOWN` at an instance, and immediately when nodes are declared `DOWN`
- Report errors to clients immediately when Oracle detects the `NOT RESTARTING` state, instead of making the client wait while the service repeatedly attempts to restart
- Balance work requests at run time using load balancing advisory events

The next sections describe how to enable FAN events for the several specific client development environments:

- Enabling JDBC Clients for Fast Connection Failover
- Enabling Oracle Call Interface Clients for Fast Connection Failover
- Enabling Oracle Call Interface Clients for Runtime Connection Load Balancing
- Enabling ODP.NET Clients to Receive FAN High Availability Events
- Enabling ODP.NET Clients to Receive FAN Load Balancing Advisory Events

Enabling JDBC Clients for Fast Connection Failover

Enabling FAN for the Oracle JDBC Implicit Connection Cache enables FAN high availability events and the load balancing advisory. Your application can use the JDBC development environment for either thick or thin JDBC clients to use FAN. You must use the JDBC Implicit Connection Cache to enable the FAN features of Fast Connection Failover and runtime connection load balancing.

To configure the JDBC client, set the `FastConnectionFailoverEnabled` property before making the first `getConnection()` request to a data source. When you enable Fast Connection Failover, the failover applies to every connection in the connection cache. If your application explicitly creates a connection cache using the Connection Cache Manager, then you must first set `FastConnectionFailoverEnabled`.

> **See Also:** The *Oracle Database JDBC Developer's Guide and Reference* chapter about "Implicit Connection Caching" for more information about configuring the JDBC implicit connection cache and ONS

Using FAN with Thick JDBC and Thin JDBC Clients

This procedure explains how to enable FAN events for JDBC. For thick JDBC clients, if you enable Fast Connection Failover, do not enable TAF, either on the client or for the service. Enabling Fast Connection Failover with thin or thick JDBC clients enables the connection pool to receive and react to all FAN events. To enable Fast Connection Failover, you must first enable the Implicit Connection Cache as described in the following procedure:

1. On a cache enabled DataSource, set the DataSource property `FastConnectionFailoverEnabled` to `true` as in the following example to enable FAN for Oracle implicit JDBC connect cache:

```
OracleDataSource ods = new OracleDataSource()
ods.setUser("Scott");
```

```
...
ods.setPassword("tigercat");
ods.setConnectionCachingEnabled(True);
ods.setFastConnectionFailoverEnabled(True);
ods.setConnectionCacheName("MyCache");
ods.setConnectionCacheProperties(cp);
ods.setURL("jdbc:oracle:thin:@(DESCRIPTION=
   (LOAD_BALANCE=on)
   (ADDRESS=(PROTOCOL=TCP)(HOST=VIP1)(PORT=1521))
   (ADDRESS=(PROTOCOL=TCP)(HOST=VIP2)(PORT=1521))
   (CONNECT_DATA=(service_name=service_name)))");
```

Note: Use the following system property to enable FAN without making data source changes: `-D oracle.jdbc.FastConnectionFailover=true`.

2. Ensure that you have configured ONS on each node that is running Oracle Clusterware as in the following example using the *CRS_home*`/opmn/conf/ons.config` file.

Note: This should have been automatically completed during the Oracle RAC installation.

```
localport=6100 # This is the port ONS is writing to on this node
remoteport=6200 # This is the port ONS is listening on this node
loglevel=3
useocr=on
```

The information in the Oracle Cluster Registry (OCR) for ONS daemons is automatically configured by DBCA. If you are upgrading from a previous version of the database, then manually configure the OCR as follows. To add the middle tier nodes or to update the Oracle RAC nodes, use `racgons` from the Oracle Clusterware `bin` directory:

- To add the ONS daemon configuration:

```
racgons add_config hostname:port [hostname:port] ...
```

- To remove the ONS daemon configuration:

```
racgons remove_config hostname[:port] [hostname:port]..
```

3. Configure remote ONS subscription. Remote ONS subscription offers the following advantages:
   - Support for an All Java mid-tier software
   - An ONS daemon is not needed on the client system you do not need to manage this process
   - Simple configuration by way of a `DataSource` property

   When using remote ONS subscription for Fast Connection Failover, an application uses `setONSConfiguration`, using the string `remoteONSConfig`, on an Oracle DataSource instance as in the following example:

```
ods.setONSConfiguration("nodes=racnode1:4200,racnode2:4200");
```

4. When you start the application, make sure that the `ons.jar` file is located on the application `CLASSPATH`. The `ons.jar` file is part of the Oracle client installation.

   See Also: *Oracle Database JDBC Developer's Guide and Reference* for more information about JDBC

Enabling Oracle Call Interface Clients for Fast Connection Failover

OCI clients can enable Fast Connection Failover by registering to receive notifications about Oracle RAC high availability FAN events and respond when events occur. This improves the session failover response time in OCI and also removes terminated connections from connection and session pools. This feature works on OCI applications, including those that use TAF, connection pools, or session pools.

First, you must enable a service for high availability events. This automatically populates the advanced queuing `ALERT_QUEUE`. If your application is using TAF, then enable the TAF settings for the service. Configure client applications to connect to an Oracle RAC database. Clients can register callbacks that are used whenever an event occurs. This reduces the time that it takes to detect a connection failure. During `DOWN` event processing, OCI:

- Terminates affected connections at the client and returns an error
- Removes connections from the OCI connection pool and the OCI session pool—the session pool maps each session to a physical connection in the connection pool, and there can be multiple sessions for each connection
- Fails over the connection if you have configured TAF

If TAF is not configured, then the client only receives an error.

Note: OCI does not manage `UP` events.

Perform the following steps to configure Fast Connection Failover with an OCI client:

1. Ensure that the service that you are using has Advanced Queuing notifications enabled by setting the services' values of `AQ_HA_NOTIFICATIONS` to `TRUE`. For example:

```
EXECUTE DBMS_SERVICE.MODIFY_SERVICE (service_name => 'gl.us.oracle.com'
        , aq_ha_notifications => TRUE
        , failover_method  => DBMS_SERVICE.FAILOVER_METHOD_BASIC
        , failover_type    => DBMS_SERVICE.FAILOVER_TYPE_SELECT
        , failover_retries => 180
        , failover_delay   => 5
        , clb_goal => DBMS_SERVICE.CLB_GOAL_LONG);
```

2. Enable `OCI_EVENTS` at environment creation time on the client as follows:

```
( OCIEnvCreate(...) )
```

3. Client applications must link with the client thread or operating system library.
4. Optionally register a client `EVENT` callback

To see the alert information, you can query the views `DBA_OUTSTANDING_ALERTS` and `DBA_ALERT_HISTORY`.

See Also: *Oracle Call Interface Programmer's Guide* for more information about OCI and the *Oracle Database Net Services Administrator's Guide* for more information about configuring TAF

Enabling Oracle Call Interface Clients for Runtime Connection Load Balancing

As of Oracle Database 11*g* Release 1 (11.1), Oracle Call Interface (OCI) session pooling enables multiple threads of an application to use a dynamically managed set of pre-created database sessions. Oracle continually reuses the sessions in the pool to form nearly permanent channels to the instances, thus saving the overhead of creating and closing sessions every time applications need them.

To fully optimize session pools with Oracle RAC, sessions given to the requesting threads are appropriately mapped to instances based on the instance load. The OCI session pools use service metrics information received from the Load Balancing Advisory in Oracle RAC to allocate the sessions, resulting in better performance

OCI dynamically maps sessions to the threads based on the instance load. It uses service metrics provided by the Load Balancing Advisory to perform runtime connection load balancing.[1] The number of connections can exceed the number of threads at certain times, but the OCI eventually adjusts this automatically, without any need for user intervention.

If the session pool supports services that span multiple instances, then the work requests are distributed across instances so that the instances providing better service are given more requests. For session pools that support services at only one instance, the first available session in the pool is adequate.

Runtime connection load balancing is enabled by default in a Oracle Database Release 11.1 (or higher) client talking to a server running Oracle Database Release 10.2 (or higher). To disable runtime connection load balancing, set the mode parameter to `OCI_SPC_NO_RLB` when calling `OCISessionPoolCreate()`.

To receive the service metrics based on the service time, ensure you have met the following conditions:

1. Link the application with the threads library.
2. Create the OCI environment in `OCI_EVENTS` and `OCI_THREADED` mode.
3. Enable event notification for the service for which the session pool is created.

   To enable event notification, use the `DBMS_SERVICE.MODIFY_SERVICE` procedure and set the Advanced Queuing `AQ_HA_NOTIFICATIONS` parameter to `TRUE`.
4. Modify the service to set up its service goal and `clb_goal`, as shown in the following example for the `myService` service:

```
EXEC DBMS_SERVICE.MODIFY_SERVICE(''myService'',
DBMS_SERVICE.GOAL_SERVICE_TIME,
DBMS_SERVICE.CLB_GOAL=SHORT);
```

Enabling ODP.NET Clients to Receive FAN High Availability Events

Oracle Data Provider for .NET (ODP.NET) connection pools can subscribe to notifications that indicate when nodes, services, and service members are down. After

[1] Runtime connection load balancing is basically routing work requests to sessions in a session pool that can best serve the work. It comes into effect when selecting a session from an existing session pool. Thus, runtime connection load balancing is a very frequent activity.

a DOWN event, Oracle Database cleans up sessions in the connection pool that go to the instance that stops and ODP.NET proactively disposes connections that are no longer valid. ODP.NET establishes connections to existing Oracle RAC instances if the removal of severed connections brings the total number of connections below the value that is set for the MIN_POOL_SIZE parameter.

The procedures for enabling ODP.NET are similar to the procedures for enabling JDBC in that you must set parameters in the connection string to enable Fast Connection Failover. Perform the following steps to enable FAN:

1. Enable Advanced Queuing notifications by using the DBMS.SERVICE package as in the following example:

```
EXECUTE DBMS_SERVICE.MODIFY_SERVICE (service_name => 'gl.us.oracle.com'
        , aq_ha_notifications => TRUE
        , clb_goal => DBMS_SERVICE.CLB_GOAL_LONG);
```

2. Enable Fast Connection Failover for ODP.NET connection pools by subscribing to FAN high availability events. Do this by setting the ha events connection string attribute to true at connection time. Note that this only works if you are using connection pools. In other words, do this if you have set the pooling attribute to true, which is the default. The following example shows this in more detail:

```
// C#
using System;
using Oracle.DataAccess.Client;

class HAEventEnablingSample
{
  static void Main()
  {
    OracleConnection con = new OracleConnection();

    // Open a connection using ConnectionString attributes
    // Also, enable "load balancing"
    con.ConnectionString =
      "User Id=scott;Password=tigercat;Data Source=oracle;" +
      "Min Pool Size=10;Connection Lifetime=120;Connection Timeout=60;" +
      "HA Events=true;Incr Pool Size=5;Decr Pool Size=2";

    con.Open();

    // Create more connections and carry out work against the DB here.

    // Dispose OracleConnection object
    con.Dispose();
  }
}
```

Enabling ODP.NET Clients to Receive FAN Load Balancing Advisory Events

Use the following procedures to enable ODP.NET to receive FAN load balancing advisory events:

1. Enable Advanced Queuing notifications by using the DBMS.SERVICE package, as in the following example:

```
EXECUTE DBMS_SERVICE.MODIFY_SERVICE (service_name => 'gl.us.oracle.com'
        , aq_ha_notifications => TRUE
        , failover_method  => DBMS_SERVICE.FAILOVER_METHOD_BASIC
```

```
        , failover_type    => DBMS_SERVICE.FAILOVER_TYPE_SELECT
        , failover_retries => 180
        , failover_delay   => 5
        , clb_goal => DBMS_SERVICE.CLB_GOAL_LONG);
```

2. Set the `GOAL` and `CLB_GOAL` for the service and ensure that Oracle Net Services is configured for connection load balancing.
3. To take advantage of load balancing events with ODP.NET connection pools, set the load balancing string to `TRUE`; the default is `FALSE`. You can do this at connect time. This only works if you are using connection pools, or when the pooling attribute is set to `TRUE` which is the default, as in the following example:

```
// C#
using System;
using Oracle.DataAccess.Client;

class LoadBalancingEnablingSample
{
  static void Main()
  {
    OracleConnection con = new OracleConnection();

    // Open a connection using ConnectionString attributes
    // Also, enable "load balancing"
    con.ConnectionString =
      "User Id=scott;Password=tigercat;Data Source=oracle;" +
      "Min Pool Size=10;Connection Lifetime=120;Connection Timeout=60;" +
      "Load Balancing=true;Incr Pool Size=5;Decr Pool Size=2";

    con.Open();

    // Create more connections and carry out work against the DB here.

    // Dispose OracleConnection object
    con.Dispose();
  }
}
```

See Also: *Oracle Data Provider for .NET Developer's Guide* for more information about ODP. NET and the *Oracle Database PL/SQL Packages and Types Reference* for more information about the `DBMS_SERVICES` PL/SQL package

Note: ODP.NET does not support connection re-distribution when a node starts. However, if you have enabled failover on the server-side, then ODP.NET can migrate connections to available instances.

Enabling Event Notification for Connection Failures in Oracle Real Application Clusters

Event notification is enabled if the `SQL_ORCLATTR_FAILOVER_CALLBACK` and `SQL_ORCLATTR_FAILOVER_HANDLE` attributes of the `SQLSetConnectAttr` function are set when a connection failure occurs in an Oracle RAC database environment. The symbols for the new attributes are defined in the `sqora.h` file. The `SQL_ORCLATTR_`

FAILOVER_CALLBACK attribute is used to specify the address of a routine to call when a failure event takes place.

The SQL_ORCLATTR_FAILOVER_HANDLE attribute is used to specify a context handle that will be passed as one of the parameters in the callback routine. This attribute is necessary for the ODBC application to determine which connection the failure event is taking place on.

The function prototype for the callback routine is as follows:

```
void failover_callback(void *handle, SQLINTEGER fo_code)
```

The handle parameter is the value that was set by the SQL_ORCLATTR_FAILOVER_HANDLE attribute. Null is returned if the attribute has not been set.

The fo_code parameter identifies the failure event that is taking place. The failure events map directly to the events defined in the OCI programming interface. The list of possible events is as follows:

- ODBC_FO_BEGIN
- ODBC_FO_ERROR
- ODBC_FO_ABORT
- ODBC_FO_REAUTH
- ODBC_FO_END

The following is a sample program that demonstrates how to use this feature:

```
/*
NAME
ODBCCallbackTest
DESCRIPTION
Program to demonstrate the connection failover callback feature.
PUBLIC FUNCTION(S)
main
PRIVATE FUNCTION(S)
NOTES
Command Line: ODBCCallbackTest filename [odbc-driver]
*/
#include <malloc.h>
#include <stdio.h>
#include <string.h>
#include <sql.h>
#include <sqlext.h>
#include <sqora.h>
#define TRUE  1
#define FALSE 0
/*
 * Funtion Prototypes
 */
void display_errors(SQLSMALLINT HandleType, SQLHANDLE Handle);
void failover_callback(void *Handle, SQLINTEGER fo_code);
/*
 * Macros
 */
#define ODBC_STS_CHECK(sts) \
if (sts != SQL_SUCCESS) \
{ \
display_errors(SQL_HANDLE_ENV, hEnv); \
display_errors(SQL_HANDLE_DBC, hDbc); \
display_errors(SQL_HANDLE_STMT, hStmt); \
```

```
return FALSE; \
}
/*
 * ODBC Handles
 */
SQLHENV *hEnv = NULL; // ODBC Environment Handle
SQLHANDLE *hDbc = NULL; // ODBC Connection Handle
SQLHANDLE *hStmt = NULL; // ODBC Statement Handle
/*
 * MAIN Routine
 */
main(int argc, char **argv)
{
SQLRETURN rc;
/*
 * Connection Information
 */
SQLTCHAR *dsn = "odbctest";
SQLTCHAR *uid = "scott";
SQLTCHAR *pwd = "tiger";
SQLTCHAR *szSelect = "select * from emp";
/*
 * Allocate handles
 */
rc = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, (SQLHANDLE *)&hEnv);
ODBC_STS_CHECK(rc)
rc = SQLSetEnvAttr(hEnv, SQL_ATTR_ODBC_VERSION, (SQLPOINTER)SQL_OV_ODBC3, 0);
ODBC_STS_CHECK(rc);
rc = SQLAllocHandle(SQL_HANDLE_DBC, hEnv, (SQLHANDLE *)&hDbc);
ODBC_STS_CHECK(rc);
/*
 * Connect to the database
 */
rc = SQLConnect(hDbc, dsn, (SQLSMALLINT)strlen(dsn),
uid, (SQLSMALLINT)strlen(uid),
pwd, (SQLSMALLINT)strlen(pwd));
ODBC_STS_CHECK(rc);
/*
 * Set the connection failover attributes
 */
rc = SQLSetConnectAttr(hDbc, SQL_ORCLATTR_FAILOVER_CALLBACK, &failover_
callback,0);
ODBC_STS_CHECK(rc);
rc = SQLSetConnectAttr(hDbc, SQL_ORCLATTR_FAILOVER_HANDLE, hDbc, 0);
ODBC_STS_CHECK(rc);
/*
 * Allocate the statement handle
 */
rc = SQLAllocHandle(SQL_HANDLE_STMT, hDbc, (SQLHANDLE *)&hStmt);
ODBC_STS_CHECK(rc);
/*
 * Wait for connection failovers
 */
while (TRUE)
{
  sleep(5000);
  rc = SQLExecDirect(hStmt,szSelect, strlen(szSelect));
  ODBC_STS_CHECK(rc);
  rc = SQLFreeStmt(hStmt, SQL_CLOSE);
  ODBC_STS_CHECK(rc);
```

```
}
/*
 * Free up the handles and close the connection
 */
rc = SQLFreeHandle(SQL_HANDLE_STMT, hStmt);
ODBC_STS_CHECK(rc);
rc = SQLDisconnect(hDbc);
ODBC_STS_CHECK(rc);
rc = SQLFreeHandle(SQL_HANDLE_DBC, hDbc);
ODBC_STS_CHECK(rc);
rc = SQLFreeHandle(SQL_HANDLE_ENV, hEnv);
ODBC_STS_CHECK(rc);
return TRUE;
}
/*
 * Failover Callback Routine
 */
void failover_callback(void *Handle, SQLINTEGER fo_code)
{
switch (fo_code)
{
case ODBC_FO_BEGIN:
    printf("ODBC_FO_BEGIN recevied");
        break;
case ODBC_FO_ERROR:
    printf("ODBC_FO_ERROR recevied");
        break;
case ODBC_FO_ABORT:
        printf("ODBC_FO_ABORT recevied");
        break;
case ODBC_FO_REAUTH:
        printf("ODBC_FO_REAUTH recevied");
        break;
case ODBC_FO_END:
        printf("ODBC_FO_END recevied");
        break;
default:
        printf("Invalid or unknown ODBC failover code recevied");
        break;
};
return;
}
/*
 * Retrieve the errors associated with the handle passed
 * and display them.
 */
void display_errors(SQLSMALLINT HandleType, SQLHANDLE Handle)
{
SQLTCHAR MessageText[256];
SQLTCHAR SqlState[5+1];
SQLSMALLINT i=1;
SQLINTEGER NativeError;
SQLSMALLINT TextLength;
SQLRETURN sts = SQL_SUCCESS;
if (Handle == NULL) return;
/*
 * Make sure all SQLState text is null terminated
 */
SqlState[5] = '\0';
/*
```

```
 * Fetch and display all diagnostic records that exist for this handle
 */
while (sts == SQL_SUCCESS)
{
NativeError = 0;
TextLength = 0;
sts = SQLGetDiagRec(HandleType, Handle, i, SqlState, &NativeError,
(SQLTCHAR *)&MessageText, sizeof(MessageText),&TextLength);
if (sts == SQL_SUCCESS)
{
  printf("[%s]%s\n",NativeError, &MessageText);
  if (NativeError != 0)
  {
    printf("Native Error Code: %d", NativeError);
  }
  i++;
}
}
return;
}
```

Services and Distributed Transaction Processing in Oracle Real Application Clusters

An XA transaction can span Oracle RAC instances by default, allowing any application that uses XA to take full advantage of the Oracle RAC environment to enhance the availability and scalability of the application.

This feature is controlled through the `GLOBAL_TXN_PROCESSES` initialization parameter, which is set to `1` by default. This parameter specifies the initial number of GTX*n* background processes for each Oracle RAC instance. Keep this parameter at its default value clusterwide to allow distributed transactions to span more than one Oracle RAC instance.

This allows the units of work performed across these Oracle RAC instances to share resources and act as a single transaction (that is, the units of work are tightly coupled). It also allows 2PC requests to be sent to any node in the cluster. Tightly coupled XA transactions no longer require the special type of singleton services (that is, Oracle Distributed Transaction Processing (DTP) services) to be deployed on Oracle RAC database. XA transactions are transparently supported on Oracle RAC databases with any type of services configuration.

> **See Also:** *Oracle Database Advanced Application Developer's Guide* for complete information about using Oracle XA with Oracle RAC, and *Oracle Database Reference* for information about the `GLOBAL_TXN_PROCESSES` initialization parameter

To provide improved application performance with distributed transaction processing in Oracle RAC, you may want to take advantage of the specialized service referred to as a DTP Service. Using DTP services, you can direct all branches of a distributed transaction to a single instance in the cluster. To load balance across the cluster, it is better to have several groups of smaller application servers with each group directing its transactions to a single service, or set of services, than to have one or two larger application servers.

In addition, connection pools at the application server tier that load balance across multiple connections to an Oracle RAC database can use this method to ensure that all

tightly-coupled branches of a global distributed transaction run on only one Oracle RAC instance. This is also true in distributed transaction environments using protocols such as X/Open Distributed Transaction Processing (DTP) or the Microsoft Distributed Transaction Coordinator (DTC).

To enhance the performance of distributed transactions, you can use services to manage DTP environments. By defining the DTP property of a service, the service is guaranteed to run on one instance at a time in an Oracle RAC database. All global distributed transactions performed through the DTP service are ensured to have their tightly-coupled branches running on a single Oracle RAC instance. This has the following benefits:

- The changes are available locally within one Oracle RAC instance when tightly coupled branches need information about changes made by each other
- Relocation and failover of services are fully supported for DTP
- By using more DTP services than there are Oracle RAC instances, Oracle can balance the load by services across all of the Oracle RAC database instances

To leverage all of the instances in a cluster, create one or more DTP services for each Oracle RAC instance that hosts distributed transactions. Choose one DTP service for one distributed transaction. Choose different DTP services for different distributed transactions to balance the workload among the Oracle RAC database instances. Because all of the branches of a distributed transaction are on one instance, you can leverage all of the instances to balance the load of many DTP transactions through multiple singleton services, thereby maximizing application throughput.

An external transaction manager, such as OraMTS, coordinates DTP/XA transactions. However, an internal Oracle transaction manager coordinates distributed SQL transactions. Both DTP/XA and distributed SQL transactions must use the DTP service in Oracle RAC.

If you add or delete nodes from your cluster database, then you may need to identify and relocate services that you are using for DTP transactions to ensure that you maintain optimum performance levels.

> **See Also:** *Oracle Database Advanced Application Developer's Guide* for more information about transaction branch management in Oracle RAC

Enabling Distributed Transaction Processing for Services

For services that you are going to use for distributed transaction processing, create the service using Enterprise Manager, or `SRVCTL` and define only one instance as the preferred instance. You can have as many `AVAILABLE` instances as you want. For example, the following `SRVCTL` command creates a singleton service for database `crm, xa_01.service.us.oracle.com`, whose preferred instance is RAC01:

```
srvctl add service -d crm -s xa_01.service.us.oracle.com -r RAC01 -a RAC02, RAC03
```

Then mark the service for distributed transaction processing by setting the `DTP` parameter to `TRUE`; the default is `FALSE`. Enterprise Manager enables you to set this parameter on the Cluster Managed Database Services: Create Service or Modify Service page. You can also use the `DBMS_SERVICE` package to modify the DTP property of the singleton service, as follows:

```
EXECUTE DBMS_SERVICE.MODIFY_SERVICE(service_name =>'xa_01.service.us.oracle.com',
DTP=>TRUE);
```

If, for example, RAC01 (that provides service XA_01) fails, then the singleton service that it provided fails over to one of the other instances, such as RAC02 or RAC03.

If services migrate to other instances after the cold-start of the Oracle RAC database, then you might need to force the relocation of the service to evenly re-balance the load on all of the available hardware. Use data from the `GV$ACTIVE_SERVICES` view to determine whether to do this.

Administering Services

When you create and administer services, you are dividing the work that your database performs into manageable units. The goal of using services is to achieve optimal utilization of your database infrastructure. You can create and deploy services based on business requirements and Oracle Database can measure the performance for each service. You can define both the application modules within a service as well as the individual actions for a module and monitor thresholds for these actions. This enables you to manage workloads to deliver capacity on demand.

When you create new services for your database, you should define each service's automatic workload management characteristics. The characteristics of a service include:

- A unique global name to identify the service.
- An Oracle Net Services name that a client uses to connect to the service.
- The preferred instances where you want the service to be enabled in a normal running environment.
- The available instances where the service can be enabled if one of the preferred instances fails.
- A service goal that determines whether connections are made to the service based on best service quality (how efficiently a single transaction completes) or best throughput (how efficiently a complete job or long-running query completes), as determined by the load balancing advisory.
- An indicator that determines whether the service is used for distributed transactions.
- An indicator that determines whether Oracle RAC high availability events are sent to OCI and ODP.NET clients that have registered to receive them through Advanced Queuing.

  See Also: "Enabling Oracle Call Interface Clients for Fast Connection Failover" on page 4-15 for more details

- The characteristics of session failovers such as whether failovers occur, whether sessions can use pre-existing connections on the failover instance, and whether failed over sessions can continue to process interrupted queries. The service definition can also define the number of times that a failed session attempts to reconnect to the service and how long it should wait between reconnection attempts. The service definition can also include a connection load balancing goal that informs the listener how to balance connection requests across the instances that provide the service.
- The method for load balancing connections for each service. This method is used by the listener when Oracle creates connections. Connections are classified as `LONG` (such as connection pools and SQL*FORMS) which tells the listener to use session based, or `SHORT` which tells the listener to use CPU based. If load

balancing advisory is enabled, its information is used to balance connections otherwise CPU utilization is used.

In addition to creating services, you may need to:

- Delete a service. You can delete services that you created. However, you cannot delete or modify the properties of the default database service that Oracle Database created.
- Check the status of a service. A service can be assigned different roles among the available instances. In a complex database with many services, you may not remember the details of every service. Therefore, you may need to check the status on an instance or service basis. For example, you may need to know the status of a service for a particular instance before you make modifications to that instance or to the Oracle home from which it runs.
- Start or stop a service for a database or an instance. A service must be started before it can be used for client connections to that instance. If you shutdown your database, for example, by running the Server Control (`SRVCTL`) command `srvctl stop database -d` *`database_name`* where *`database_name`* is the name of the database that you want to stop, then Oracle Database stops all services to that database. You must manually restart the services when you start the database.
- Map a Service to a Consumer Group—Oracle Database can automatically map services to Resource Manager Consumer groups to limit the amount of resources that services can use in an instance. You must create the consumer group and then map the service to the consumer group.

  See Also: *Oracle Database PL/SQL Packages and Types Reference* for information about the `DBMS_RESOURCE_MANAGER.SET_CONSUMER_GROUP_MAPPING_PRI` procedure

- Enable or disable a service for a database or an instance. By default, Oracle Clusterware attempts to restart a service automatically after failures. You can prevent this behavior by disabling a service. Disabling a service is useful when you need to perform database or instance maintenance, for example, if you are performing an upgrade and you want to prevent connection requests.
- Relocate a service to a different instance. You can move a service from one instance to another instance to re-balance workloads, for example, after adding or deleting cluster nodes.
- If you execute a SQL statement in parallel, the parallel processes only run on the instances that offer the service with which you originally connected to the database. This is the default behavior. This does not affect other parallel operations such as parallel recovery or the processing of `GV$` queries. To override this behavior, set a value for the `PARALLEL_INSTANCE_GROUPS` initialization parameter.

Notes:

- When you use services, do not set a value for the SERVICE_NAMES parameter; Oracle controls the setting for this parameter for the services that you create and for the default database service. The service features that this chapter describes are not directly related to the features that Oracle provides when you set SERVICE_NAMES. In addition, setting a value for this parameter may override some of the benefits of using services.
- Using service names to access a queue provides location transparency for the queue in an Oracle RAC database.
- If you specify a service using the DISPATCHERS initialization parameter, it overrides any service in the SERVICE_NAMES parameter, and cannot be managed. (For example, stopping the service with a SRVCTL command will not stop users connecting with the service.)

Administering Services with Enterprise Manager, PL/SQL, and SRVCTL

You can create and administer services with Enterprise Manager and the DBMS_SERVICE PL/SQL package, and you can perform most service administration tasks with the SRVCTL utility. The following sections describe how to perform service-related tasks using these tools:

- Administering Services with Enterprise Manager
- Administering Services with the PL/SQL DBMS_SERVICE Package
- Administering Services with SRVCTL

Administering Services with Enterprise Manager

The Cluster Managed Database Services page is the master page for beginning all tasks related to services. To access this page, go to the Cluster Database Availability page, then click **Cluster Managed Database Services** in the Services section. You can use this page and drill down from this page to:

- View a list of services for the cluster
- View the instances on which each service is currently running
- View the status for each service
- Create or edit a service
- Start or stop a service
- Enable or disable a service
- Perform instance-level tasks for a service
- Delete a service

See Also: *Oracle Enterprise Manager Concepts* for more information about administering services with Enterprise Manager

Service-Related Tasks That You Can Perform with Enterprise Manager

You can perform service-related tasks as described for the following Enterprise Manager pages:

- Cluster Managed Database Services Page
- Cluster Managed Database Services Detail Page
- Create Services Page

Cluster Managed Database Services Page

The Cluster Managed Database Services page enables you to:

- View a list of services for the cluster, the instances on which each service is currently running, and the status for each service
- Start or stop a service, or enable or disable a service
- Access the Create Service and Edit Service pages
- Access the Services Detail page to perform instance-level tasks for a service
- Test the connection for a service

Cluster Managed Database Services Detail Page

The Cluster Managed Database Services Detail page enables you to:

- View the status of a service on all of its preferred and available instances; the status can be `Running`, `Stopped`, or `Disabled`
- Stop or start a service for an instance of a cluster database
- Disable or enable a service for an instance of a cluster database
- Relocate a service to manually re-balance the services load

Create Services Page

The Create Services page enables you to:

- Create the service with name, high availability and performance attributes
- Select the desired service policy for each instance configured for the cluster database
- Select the desired service properties, such as the TAF policy. (This configures client-side TAF. To set the service TAF policy, use the `DBMS_SERVICE` package.) You can also set the notification properties, load balancing goals, alert levels, and resource management properties

> **See Also:** *Oracle Database Net Services Administrator's Guide* for more information about configuring TAF

Accessing the Enterprise Manager Services Pages

To access the Cluster Managed Database Services page and detail pages for service instances:

1. From the Cluster Database Home page, click the **Availability** tab.
2. From the Cluster Database Availability page, under the Services heading in the High Availability options list, click **Cluster Managed Database Services**. The Cluster and Database Login page appears.
3. Enter credentials for the database and for the cluster that hosts the cluster database and click **Continue**. The Cluster Managed Database Services page appears and displays services that are available on the cluster database instances. For information about performing tasks on this page, see the online help for this page.

Note: You must have SYSDBA credentials to access a cluster database. Cluster Managed Database Services does not permit you to connect as anything other than SYSDBA.

4. To manage a service at the instance level, either click a service name or select a service name, then select **Manage** from the Actions drop-down list and click **Go**. The Cluster Managed Database Services Detail page for the service appears. For information about performing tasks on this page, see the online help for this page.

To access the Relocate page:

1. Perform steps 1 - 3 from the previous procedure set.
2. From the Cluster Managed Database Services page, either click a service name or select a service name, then select **Manage** from the Actions drop-down list and click **Go**. The Cluster Managed Database Services Detail page for the service appears.
3. Click **Relocate**. The Relocate Service from Instance page appears, where you can perform manual load balancing of services. For information about performing tasks on this page, see the online help for this page.

Administering Services with the PL/SQL DBMS_SERVICE Package

You can manage services with the PL/SQL DBMS_SERVICE package procedures described in Table 4–4.

Note: Oracle recommends that you use Enterprise Manager to create services for Oracle RAC environments.

Table 4–4 DBMS_SERVICE Package Procedures for Administering Services

Procedure	Description
CREATE_SERVICE	Adds a new service to the Oracle RAC database. In the CREATE_SERVICE syntax: ■ service_name is the unique, global service name ■ network_name is the TNS name for connections to the service ■ goal sets the automatic workload management goal directive to service quality or throughput ■ dtp is the distributed transaction flag ■ aq_ha_notification is the flag to send Oracle RAC high availability events to registered OCI clients ■ failover_method is the TAF failover method for the service ■ failover_type is the TAF failover method for the service ■ failover_retries is the TAF connection retry count ■ failover_delay is the wait time between TAF connection retries ■ clb_goal sets the runtime connection load balancing goal When using services with Oracle RAC, use Enterprise Manager or SRVCTL commands to add the high availability properties, such as the PREFERRED and AVAILABLE placement.

Table 4–4 (Cont.) DBMS_SERVICE Package Procedures for Administering Services

Procedure	Description
MODIFY_SERVICE	Changes one or more service characteristics. The parameters for the MODIFY_SERVICE procedure are the same as those for the CREATE_SERVICE procedure.
DELETE_SERVICE	Drops an existing service. In the DELETE_SERVICE procedure, service_name is the name of the service to be dropped.
START_SERVICE	Starts a service on the connected instance, the named instance, or all instances. In the START_SERVICE syntax SERVICE_NAME is the name of the service to be started and INSTANCE_NAME can be NULL (to start the service on the connected instance), the name of an instance (to start the service on that instance), or the package constant DBMS_SERVICE.ALL_INSTANCES to start the service on all instances where it is defined.
STOP_SERVICE	Stops a service on the connected instance, the named instance, or all instances. The parameters for the STOP_SERVICE procedure are the same as those for the START_SERVICE procedure.
DISCONNECT_SESSION	Terminates all sessions on the connected instance, or the instance where the PL/SQL procedure is run. In the DISCONNECT_SESSION procedure, service_name is the name of the service for which connections are to be dropped.

See Also: *Oracle Database PL/SQL Packages and Types Reference* for more information about the CREATE_SERVICE procedure syntax and for information about other procedures mentioned in this section

Also, see the *Oracle Database Net Services Administrator's Guide* for more information about configuring TAF

Administering Services with SRVCTL

When you create a service with SRVCTL, you must start it with a separate SRVCTL command. However, you may later need to manually stop or restart the service. You may also need to disable the service to prevent automatic restarts, to manually relocate the service, or obtain status information about the service. The following sections explain how to use SRVCTL to perform the following administrative tasks:

- Creating Services with SRVCTL
- Starting and Stopping Services with SRVCTL
- Enabling and Disabling Services with SRVCTL
- Relocating Services with SRVCTL
- Obtaining the Statuses of Services with SRVCTL
- Obtaining the Configuration of Services with SRVCTL

See Also: Appendix A, "Server Control Utility Reference" for more information about SRVCTL and the other SRVCTL commands that you can use to manage services

Creating Services with SRVCTL

To create a service with SRVCTL, enter a command from the command line using the following syntax:

```
srvctl add service -d database_unique_name -s service_name -r preferred_list
```

```
[-a available_list] [-P TAF_policy]
```

Starting and Stopping Services with SRVCTL

Enter the following SRVCTL syntax from the command line:

```
srvctl start service -d database_unique_name [-s service_name_list] [-i inst_name]
[-o start_options]

srvctl stop service -d database_unique_name -s service_name_list [-i inst_name]
[-o start_options]
```

Enabling and Disabling Services with SRVCTL

Use the following SRVCTL syntax from the command line to enable and disable services:

```
srvctl enable service -d database_unique_name -s service_name_list [-i inst_name]
srvctl disable service -d database_unique_name -s service_name_list [-i inst_name]
```

Relocating Services with SRVCTL

Run the `srvctl relocate service` command from the command line to relocate a service. For example, the following command relocates the `crm` service from instance apps1 to instance apps3:

```
srvctl relocate service -d apps -s crm -i apps1 -t apps3
```

Obtaining the Statuses of Services with SRVCTL

Run the `srvctl relocate service` command from the command line to obtain the status of a service. For example, the following command returns the status of the `crm` service that is running on the crm database:

```
srvctl status service -d apps -s crm
```

Obtaining the Configuration of Services with SRVCTL

Run the `srvctl relocate service` command from the command line to obtain the high availability configuration of a service. For example, the following command returns the configuration of the `crm` service that is running on the `crm` database:

```
srvctl config service -d apps -s crm -a
```

> **See Also:** Appendix A, "Server Control Utility Reference" for information about other administrative tasks that you can perform with SRVCTL

Measuring Performance by Service Using the Automatic Workload Repository

Services add a new dimension for performance tuning. With services, workloads are visible and measurable and resource consumption and wait times are attributable by application. Tuning by using 'service and SQL' replaces tuning by 'session and SQL' in the majority of systems where all sessions are anonymous and shared.

The Automatic Workload Repository (AWR) maintains performance statistics that include information about response time, throughput, resource consumption, and wait events for all services and work that a database performs. Oracle also maintains metrics, statistics, wait events, wait classes, and SQL-level traces for services. You can optionally augment these statistics by defining modules within your application to

monitor certain statistics. You can also define the actions within those modules that business critical transactions should execute in response to particular statistical values. Enable module and action monitoring using the DBMS_MONTIOR PL/SQL package as follows:

```
EXECUTE DBMS_MONITOR.SERV_MOD_ACT_STAT_ENABLE(SERVICE_NAME => 'ERP', MODULE_NAME=>
'PAYROLL', ACTION_NAME => 'EXCEPTIONS PAY');
EXECUTE DBMS_MONITOR.SERV_MOD_ACT_STAT_ENABLE(SERVICE_NAME => 'ERP', MODULE_NAME=>
'PAYROLL', ACTION_NAME => NULL);
```

Use the DBA_ENABLED_AGGREGATIONS view to verify that you have enabled monitoring.

Statistics aggregation and tracing by service are global in scope for Oracle RAC databases. In addition, they are persistent across instance restarts and service relocations for both Oracle RAC and single-instance Oracle databases.

The service, module, and action names are visible in V$SESSION, V$ACTIVE_SESSION_HISTORY, and V$SQL views. The call times and performance statistics are visible in V$SERVICE_STATS, V$SERVICE_EVENTS, V$SERVICE_WAIT_CLASSES, V$SERVICEMETRIC, and V$SERVICEMETRIC_HISTORY. When you enable statistics collection for an important transaction, you can see the call speed for each service, module, and action name at each database instance using the V$SERV_MOD_ACT_STATS view.

The following sample SQL*Plus script provides service quality statistics every five seconds. You can use these service quality statistics to monitor the quality of a service, to direct work, and to balance services across Oracle RAC instances:

```
SET PAGESIZE 60 COLSEP '|' NUMWIDTH 8 LINESIZE 132 VERIFY OFF FEEDBACK OFF
COLUMN service_name FORMAT A20 TRUNCATED HEADING 'Service'
COLUMN begin_time HEADING 'Begin Time' FORMAT A10
COLUMN end_time HEADING 'End Time' FORMAT A10
COLUMN instance_name HEADING 'Instance' FORMAT A10
COLUMN service_time HEADING 'Service Time|mSec/Call' FORMAT 999999999
COLUMN throughput HEADING 'Calls/sec'FORMAT 99.99
BREAK ON service_name SKIP 1
SELECT
    service_name
  , TO_CHAR(begin_time, 'HH:MI:SS') begin_time
  , TO_CHAR(end_time, 'HH:MI:SS') end_time
  , instance_name
  , elapsedpercall  service_time
  ,  callspersec  throughput
FROM
    gv$instance i
  , gv$active_services s
  , gv$servicemetric m
WHERE s.inst_id = m.inst_id
  AND s.name_hash = m.service_name_hash
  AND i.inst_id = m.inst_id
  AND m.group_id = 10
ORDER BY
   service_name
 , i.inst_id
 , begin_time ;
```

Service Thresholds and Alerts

Service level thresholds enable you to compare achieved service levels against accepted minimum required levels. This provides accountability with respect to the delivery or the failure to deliver an agreed service level. The end goal is a predictable system that achieves service levels. There is no requirement to perform as fast as possible with minimum resource consumption; the requirement is to meet the *quality* of service.

You can explicitly specify two performance thresholds for each service: the response time for calls, or `SERVICE_ELAPSED_TIME`, and the CPU time for calls, or `SERVICE_CPU_TIME`. The response time goal indicates that the elapsed time should not exceed a certain value, and the response time represents wall clock time. Response time is a fundamental measure that reflects all delays and faults that might be blocking the call from running on behalf of the user. Response time can also indicate differences in node power across the nodes of an Oracle RAC database.

The service time and CPU time are calculated as the moving average of the elapsed, server-side call time. The AWR monitors the service time and CPU time and publishes AWR alerts when the performance exceeds the thresholds. You can then respond to these alerts by changing the priority of a job, stopping overloaded processes, or by relocating, expanding, shrinking, starting or stopping a service. This permits you to maintain service availability despite changes in demand.

Example of Services and Thresholds Alerts

To check the thresholds for the payroll service, use the AWR report. You should record output from the report over several successive intervals during which time the system is running optimally. For example, assume that for an e-mail server, the AWR report runs each Monday during the peak usage times of 10:00 a.m. to 2:00 p.m. The AWR report would contain the response time, or DB time, and the CPU consumption time, or CPU time, for calls for each service. The AWR report would also provide a breakdown of the work done and the wait times that are contributing to the response times.

Using `DBMS_MONITOR`, set a warning threshold for the payroll service at 0.5 seconds and a critical threshold for the payroll service at 0.75 seconds. You must set these thresholds at all instances within an Oracle RAC database. You can schedule actions using Enterprise Manager jobs for alerts, or you can schedule actions to occur programmatically when the alert is received. In this example, thresholds are added for the `servall` service and set as follows:

```
EXECUTE DBMS_SERVER_ALERT.SET_THRESHOLD(
METRICS_ID => DBMS_SERVER_ALERT.ELAPSED_TIME_PER_CALL
, warning_operator => DBMS_SERVER_ALERT.OPERATOR_GE
, warning_value => '500000'
, critical_operator => DBMS_SERVER_ALERT.OPERATOR_GE
, critical_value => '750000'
, observation_period => 30
, consecutive_occurrences => 5
, instance_name => NULL
, object_type => DBMS_SERVER_ALERT.OBJECT_TYPE_SERVICE
, object_name => 'servall');
```

Verify the threshold configuration using the following `SELECT` statement:

```
SELECT METRICS_NAME, INSTANCE_NAME, WARNING_VALUE, CRITICAL_VALUE,
OBSERVATION_PERIOD FROM dba_thresholds ;
```

Enable Service, Module, and Action Monitoring

You can enable performance data tracing for important modules and actions within each service. The performance statistics are available in the V$SERV_MOD_ACT_STATS view. As an example, set the following:

- Under the ERP service, enable monitoring for the exceptions pay action in the payroll module.
- Under the ERP service, enable monitoring for the all actions in the payroll module.
- Under the HOT_BATCH service, enable monitoring for all actions in the posting module.

```
EXECUTE DBMS_MONITOR.SERV_MOD_ACT_STAT_ENABLE(service_name => 'erp', module_name=>
 'payroll', action_name => 'exceptions pay');
EXECUTE DBMS_MONITOR.SERV_MOD_ACT_STAT_ENABLE(service_name => 'erp', module_name=>
 'payroll', action_name => NULL);
EXECUTE DBMS_MONITOR.SERV_MOD_ACT_STAT_ENABLE(service_name => 'hot_batch',
module_name =>'posting', action_name => NULL);
```

Verify the enabled service, module, action configuration with the following SELECT statement:

```
COLUMN AGGREGATION_TYPE FORMAT A21 TRUNCATED HEADING 'AGGREGATION'
COLUMN PRIMARY_ID FORMAT A20 TRUNCATED HEADING 'SERVICE'
COLUMN QUALIFIER_ID1 FORMAT A20 TRUNCATED HEADING 'MODULE'
COLUMN QUALIFIER_ID2 FORMAT A20 TRUNCATED HEADING 'ACTION'
SELECT * FROM DBA_ENABLED_AGGREGATIONS ;
```

The output might appear as follows:

```
AGGREGATION            SERVICE              MODULE        ACTION
------------           --------------------  ----------    -------------
SERVICE_MODULE_ACTION  ERP                   PAYROLL       EXCEPTIONS PAY
SERVICE_MODULE_ACTION  ERP                   PAYROLL
SERVICE_MODULE_ACTION  HOT_BATCH             POSTING
```

5

Configuring Recovery Manager and Archiving

This chapter explains how to configure Recovery Manager (RMAN) for use in Oracle Real Application Clusters (Oracle RAC) environments. This chapter also provides procedures for using RMAN for archiving in Oracle RAC environments and discusses online redo log and archived redo log considerations.

The topics in this chapter include:

- Overview of Configuring RMAN for Oracle Real Application Clusters
- Configuring the RMAN Snapshot Control File Location
- Configuring RMAN to Automatically Backup the Control File and SPFILE
- Crosschecking on Multiple Oracle Real Application Clusters Nodes
- Configuring Channels for RMAN in Oracle Real Application Clusters
- Managing Archived Redo Logs Using RMAN in Oracle Real Application Clusters
- Archived Redo Log File Conventions in Oracle Real Application Clusters
- RMAN Archiving Configuration Scenarios
- Changing the Archiving Mode in Oracle Real Application Clusters

Overview of Configuring RMAN for Oracle Real Application Clusters

RMAN enables you to back up, restore, and recover datafiles, control files, server parameter files (SPFILEs) and archived redo log files. RMAN is included with the Oracle Database and does not require separate installation. You can run RMAN from the command line or use RMAN in the Backup Manager in Oracle Enterprise Manager.

> **Tip:** When should you perform a backup? Oracle recommends performing a backup immediately after running the `root.sh` script. Also, after running the `addNode.sh` or `rootdelete.sh` scripts you should back up the voting disk (or disks).

Configuring the RMAN Snapshot Control File Location

The snapshot control file is a copy of a database control file created in an operating system-specific location by Recovery Manager. RMAN creates the snapshot control file so that it has a consistent version of a control file to use when either resynchronizing the recovery catalog or backing up the control file. In Oracle RAC, the snapshot control file is only needed on the nodes on which RMAN performs backups; the snapshot

control file does not need to be globally available to all instances in an Oracle RAC environment.

You can specify a **cluster file system** file or a raw device destination for the location of your snapshot control file. Run the following RMAN command to determine the configured location of the snapshot control file:

```
SHOW SNAPSHOT CONTROLFILE NAME;
```

You can change the configured location of the snapshot control file. For example, on Linux and UNIX systems you can specify the snapshot control file location as `$ORACLE_HOME/dbs/scf/snap_prod.cf` by entering the following at the RMAN prompt:

```
CONFIGURE SNAPSHOT CONTROLFILE NAME TO '$ORACLE_HOME/dbs/scf/snap_prod.cf';
```

This command globally sets the configuration for the location of the snapshot control file throughout your **cluster database**. Therefore, ensure that the directory `$ORACLE_HOME/dbs/scf` exists on all nodes that perform backups.

The `CONFIGURE` command creates persistent settings across RMAN sessions. Therefore, you do not need to run this command again unless you want to change the location of the snapshot control file. You can also specify a cluster file system file or a raw device destination for the location of your snapshot control file. This file is shared across all nodes in the **cluster** just like other datafiles in Oracle RAC.

> **See Also:** *Oracle Database Backup and Recovery Reference* for more information about configuring the snapshot control file

Configuring RMAN to Automatically Backup the Control File and SPFILE

If you set `CONFIGURE CONTROLFILE AUTOBACKUP` to `ON`, then RMAN automatically creates a control file and an SPFILE backup after you run the `BACKUP` or `COPY` commands. RMAN can also automatically restore an SPFILE if this is required to start an instance to perform recovery. This means that the default location for the SPFILE must be available to all nodes in your Oracle RAC database.

These features are important in disaster recovery because RMAN can restore the control file even without a recovery catalog. RMAN can restore an autobackup of the control file even after the loss of both the recovery catalog and the current control file. You can change the default name that RMAN gives to this file with the `CONFIGURE CONTROLFILE AUTOBACKUP FORMAT` command. Note that if you specify an absolute path name in this command, then this path must exist identically on all nodes that participate in backups.

RMAN performs the control file autobackup on the first allocated channel. Therefore, when you allocate multiple channels with different parameters, especially when you allocate a channel with the `CONNECT` command, determine which channel will perform the control file autobackup. Always allocate the channel for this node first.

Besides using the RMAN control file, you can also use Enterprise Manager to use the RMAN features.

> **See Also:** *Oracle Database Backup and Recovery User's Guide* for more information about using the control file autobackup feature

Crosschecking on Multiple Oracle Real Application Clusters Nodes

When crosschecking on multiple nodes (and when operating RMAN in general), configure the cluster so that all backups can be accessed by every node, regardless of which node created the backup. When the cluster is configured this way, you can allocate channels at any node in the cluster during restore or crosscheck operations.

If you cannot configure the cluster so that each node can access all backups, then during restore and crosscheck operations, you must allocate channels on multiple nodes by providing the `CONNECT` option to the `CONFIGURE CHANNEL` command, so that every backup can be accessed by at least one node. If some backups are not accessible during crosscheck because no channel was configured on the node that can access those backups, then those backups are marked `EXPIRED` in the RMAN repository after the crosscheck.

For example, you can use `CONFIGURE CHANNEL ... CONNECT` in an Oracle RAC configuration in which tape backups are created on various nodes in the cluster and each backup is only accessible on the node on which it is created. This is described in more detail in "Configuring Channels to Use a Specific Channel" on page 5-3.

Configuring Channels for RMAN in Oracle Real Application Clusters

This section describes how to configure channels for RMAN. You can configure channels to use automatic load balancing or you can specify specific channels for specific instances as described in the following topics:

- Configuring Channels to Use Automatic Load Balancing
- Configuring Channels to Use a Specific Channel

Configuring Channels to Use Automatic Load Balancing

To configure channels to use automatic load balancing, use the following syntax:

```
CONFIGURE DEVICE TYPE [disk | sbt] PARALLELISM number of channels;
...
```

Where number of channels is the *`number of channels`* that you want to use for the operation. After you complete this one-time configuration, you can issue `BACKUP` or `RESTORE` commands.

Configuring Channels to Use a Specific Channel

To configure one RMAN channel for each Oracle RAC instance, use the following syntax:

```
CONFIGURE CHANNEL DEVICE TYPE sbt CONNECT
'SYS/change_on_install@node1'
CONFIGURE CHANNEL DEVICE TYPE sbt CONNECT
'SYS/change_on_install@node2'
...
```

After this one-time configuration step, you can issue the `BACKUP` or `RESTORE` commands. In addition, you can use the `PARMS` command in this example to set vendor-specific parameters.

Managing Archived Redo Logs Using RMAN in Oracle Real Application Clusters

When a node generates an archived redo log, Oracle Database always records the filename of the log in the control file of the target database. If you are using a recovery catalog, then RMAN also records the archived redo log filenames in the recovery catalog when a resynchronization occurs.

The archived redo log naming scheme that you use is important because when a node writes to a log with a specific filename on its file system, the file must be readable by any node that needs to access this archived redo log. For example, if node 1 archives a log to `/oracle/arc_dest/log_1_100_23452345.arc`, then node 2 can only back up this archived redo log only if it can read `/oracle/arc_dest/log_1_100_23452345.arc` on its own file system.

The backup and recovery strategy that you choose depends on how you configure the archiving destinations for each node. Whether only one node or all nodes perform archived redo log backups, you need to ensure that all archived redo logs are backed up. If you use RMAN parallelism during recovery, then the node that performs recovery must have read access to all archived redo logs in your cluster.

Multiple nodes can restore archived logs in parallel. However, during recovery, one one node applies the archived logs. Therefore, the node that is performing the recovery must be able to access all of the archived logs that are needed for the recovery operation. By default, the database determines the optimum number of parallel threads to use during the recovery operation. You can use the `PARALLEL` clause in the `RECOVER` command to change this number.

Guidelines and Considerations for Archived Redo Logs

The primary consideration is to ensure that all archived redo logs can be read from every node during recovery, and if possible during backups. This section illustrates the archived redo log naming issues for configuring archiving in your cluster database. The scenario described here is for a noncluster file system archiving scheme. Assume that the following conditions are met:

- Configure each node to write to a local archiving directory that is named the same on each node.
- Do *not* set up a cluster file system (in other words, each node can only read from and write to its own local file system). See the information about cluster file systems later in this chapter.
- Do not use NFS (network file system) or mapped drives to enable the nodes in the cluster to gain read/write access to one another.

Example 5–1 Example Configuration for the initialization parameters file

```
sid1.log_archive_dest_1 = (location=/arc_dest_1)
sid2.log_archive_dest_1 = (location=/arc_dest_2)
sid3.log_archive_dest_1 = (location=/arc_dest_3)
```

Assume that the filenames of the archived redo logs are recorded in the control file as follows:

```
/arc_dest_1/log_1_62_23452345.arc
/arc_dest_2/log_2_100_23452345.arc
/arc_dest_2/log_2_101_23452345.arc
/arc_dest_3/log_3_70_23452345.arc
/arc_dest_1/log_1_63_23452345.arc
```

During recovery, as long as the archived log destinations are visible from the node that performs the recovery, Oracle Database can successfully recover the archived log data.

Archived Redo Log File Conventions in Oracle Real Application Clusters

For any archived redo log configuration, uniquely identify the archived redo logs with the LOG_ARCHIVE_FORMAT parameter. The format of this parameter is operating system-specific and it can include text strings, one or more variables, and a filename extension.

Table 5–1 Archived Redo Log Filename Format Parameters

Parameter	Description	Example
%r	Resetlogs identifier	log_1_62_23452345
%R	Padded resetlogs identifier	log_1_62_0023452345
%s	Log sequence number, not padded	log_251
%S	Log sequence number, left-zero-padded	log_0000000251
%t	Thread number, not padded	log_1
%T	Thread number, left-zero-padded	log_0001

All of the thread parameters, in either upper or lower case, are mandatory for Oracle RAC. This enables Oracle Database to create unique names for archive logs across the incarnation. This requirement is in effect when the COMPATIBLE parameter is set to 10.0 or greater.

Use the %R or %r parameters to include the resetlogs identifier to avoid overwriting the logs from a previous incarnation. If you do not specify a log format, then the default is operating system-specific and includes %t, %s, and %r.

As an example, if the instance associated with redo thread number 1 sets LOG_ARCHIVE_FORMAT to log_%t_%s_%r.arc, then its archived redo log files are named:

```
log_1_1000_23435343.arc
log_1_1001_23452345.arc
log_1_1002_23452345.arc
...
```

> **See Also:** *Oracle Database Administrator's Guide* about specifying the archived redo log filename format and destination, and Oracle Database platform-specific documentation about the default log archiving format and destination

RMAN Archiving Configuration Scenarios

This section describes the archiving scenarios for an Oracle RAC database. The two configuration scenarios in this chapter describe a three-node UNIX cluster for an Oracle RAC database. For both scenarios, the LOG_ARCHIVE_FORMAT that you specify for the instance performing recovery must be the same as the format that you specified for the instances that archived the files.

Automatic Storage Management and Cluster File System Archiving Scheme

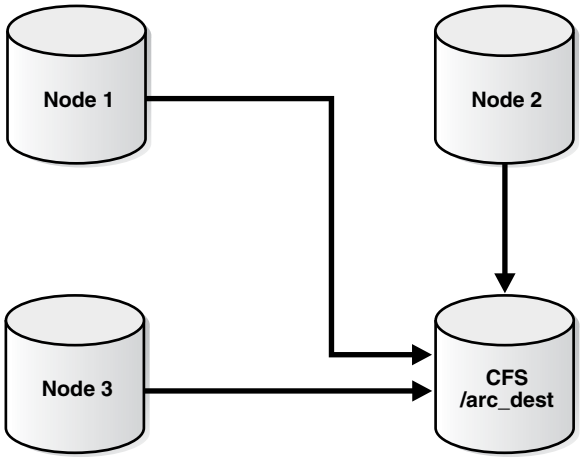
The preferred configuration for Oracle RAC is to use Automatic Storage Management (ASM) for a recovery area with a different disk group for your recovery set than for your datafiles. Alternatively, you can use a cluster file system archiving scheme.

In this case, each node writes to a single cluster file system archived redo log destination and can read the archived redo log files of the other nodes. Read access is achieved for all nodes with a cluster file system. For example, if node 1 archives a log to `/arc_dest/log_1_100_23452345.arc` on the cluster file system, then any other node in the cluster can also read this file.

> **Note:** The archive log naming format in this example is only for a CFS example. ASM uses an Oracle Managed Files (OMF)-based naming format.

If you do not use a cluster file system, then the archived redo log files cannot be on raw devices. This is because raw devices do not enable sequential writing of consecutive archive log files.

Figure 5–1 Cluster File System Archiving Scheme



Advantages of the Cluster File System Archiving Scheme

The advantage of this scheme is that none of the nodes uses the network to archive logs. Because the filename written by a node can be read by any node in the cluster, RMAN can back up all logs from any node in the cluster. Backup and restore scripts are simplified because each node has access to all archived redo logs.

Initialization Parameter Settings for the Cluster File System Archiving Scheme

In the cluster file system scheme, each node archives to a directory that is identified with the same name on all instances within the cluster database. To configure this, set values for the `LOG_ARCH_DEST_`*n* parameter for each instance using the `sid` designator as in the following example:

```
sid1.LOG_ARCHIVE_DEST_1="LOCATION=/arc_dest"
sid2.LOG_ARCHIVE_DEST_1="LOCATION=/arc_dest"
sid3.LOG_ARCHIVE_DEST_1="LOCATION=/arc_dest"
```

The following list shows archived redo log entry examples that would appear in the RMAN catalog or the in the control file based on the previous example. Note that any node can archive logs using any of the threads:

```
/arc_dest/log_1_999_23452345.arc
/arc_dest/log_1_1000_23435343.arc
/arc_dest/log_1_1001_23452345.arc <- thread 1 archived in node 3
/arc_dest/log_3_1563_23452345.arc <- thread 3 archived in node 2
/arc_dest/log_2_753_23452345.arc <- thread 2 archived in node 1
/arc_dest/log_2_754_23452345.arc
/arc_dest/log_3_1564_23452345.arc
```

Location of Archived Logs for the Cluster File System Archiving Scheme

Because the file system is shared and because each node is writing its archived redo logs to directory `/arc_dest` in the cluster file system, each node can read the logs written by itself as well as any other node.

Noncluster File System Local Archiving Scheme

In the noncluster file system local archiving scheme, each node archives to a uniquely named local directory. If recovery is required, then you can configure the recovery node so that it can access directories on the other nodes remotely. For example, use NFS on Linux and UNIX computers, or mapped drives on Windows systems. Therefore, each node writes only to a local destination, but each node can also read archived redo log files in remote directories on the other nodes.

Considerations for Using Noncluster File System Local Archiving

If you use noncluster file system local archiving for media recovery, then you must configure the node that is performing recovery for remote access to the other nodes so that it can read the archived redo log files in the archiving directories on the other nodes. In addition, if you are in recovery and if you do not have all of the available archive logs, then you must perform an incomplete recovery up to the first missing archived redo log sequence number. You do not have to use a specific configuration for this scheme. However, if you want to distribute the backup processing onto multiple nodes, then the easiest method is to configure channels as described in the backup scenarios in Chapter 6, "Managing Backup and Recovery".

Initialization Parameter Settings for Noncluster File System Local Archiving

You can set the archiving destination values as follows in the initialization parameter file:

```
sid1.LOG_ARCHIVE_DEST_1="LOCATION=/arc_dest_1"
sid2.LOG_ARCHIVE_DEST_1="LOCATION=/arc_dest_2"
sid3.LOG_ARCHIVE_DEST_1="LOCATION=/arc_dest_3"
```

The following list shows the possible archived redo log entries in the database control file. Note that any node is able to archive logs from any of the threads:

```
/arc_dest_1/log_1_1000_23435343.arc
/arc_dest_2/log_1_1001_23452345.arc <- thread 1 archived in node 2
/arc_dest_2/log_3_1563_23452345.arc <- thread 3 archived in node 2
/arc_dest_1/log_2_753_23452345.arc <- thread 2 archived in node 1
/arc_dest_2/log_2_754_23452345.arc
/arc_dest_3/log_3_1564_23452345.arc
```

Location of Archived Logs for Noncluster File System Local Archiving

As illustrated in Table 5–2, each node has a directory containing the locally archived redo logs. Additionally, if you mount directories on the other nodes remotely through NFS or mapped drives, then each node has two remote directories through which RMAN can read the archived redo log files that are archived by the remaining nodes.

Table 5–2 UNIX/NFS Location Log Examples, Noncluster File System Local Archiving

Node ...	Can read the archived redo log files in the directory ...	For logs archived by node ...
1	/arc_dest_1	1
1	/arc_dest_2	2 (through NFS)
1	/arc_dest_3	3 (through NFS)
2	/arc_dest_1	1 (through NFS)
2	/arc_dest_2	2
2	/arc_dest_3	3 (through NFS)
3	/arc_dest_1	1 (through NFS)
3	/arc_dest_2	2 (through NFS)
3	/arc_dest_3	3

File System Configuration for Noncluster File System Local Archiving

If you are performing recovery and a surviving instance must read all of the logs that are on disk but not yet backed up, then you should configure NFS as shown in Table 5–3.

Table 5–3 UNIX/NFS Configuration for Shared Read Local Archiving Examples

Node	Directory ...	Is configured ...	And mounted on ...	On node ...
1	/arc_dest_1	Local read/write	n/a	n/a
1	/arc_dest_2	NFS read	/arc_dest_2	2
1	/arc_dest_3	NFS read	/arc_dest_3	3
2	/arc_dest_1	NFS read	/arc_dest_1	1
2	/arc_dest_2	Local read/write	n/a	n/a
2	/arc_dest_3	NFS read	/arc_dest_3	3
3	/arc_dest_1	NFS read	/arc_dest_1	1
3	/arc_dest_2	NFS read	/arc_dest_2	2
3	/arc_dest_3	Local read/write	n/a	n/a

Note: Windows users can achieve the same results depicted in the examples in this section by using mapped drives.

Changing the Archiving Mode in Oracle Real Application Clusters

You can run the `ALTER DATABASE` SQL statement to change the archiving mode in Oracle RAC as long as the database is mounted by the local instance but not open in any instances. You do not need to modify parameter settings to run this statement.

Note: You can also change the archive log mode by using the Recovery Settings page in the **Maintenance** tab of the Enterprise Manager Oracle RAC Database Home Page.

Monitoring the Archiver Processes

After your RMAN configuration is operative in your Oracle RAC environment, use the `GV$ARCHIVE_PROCESSES` and `V$ARCHIVE_PROCESSES` views to determine the status of the archiver processes. Depending on whether you query the global or local views, these views display information for all database instances, or for only the instance to which you are connected.

See Also: *Oracle Database Reference* for more information about the database views

6

Managing Backup and Recovery

This chapter explains instance recovery and how to use Recovery Manager (RMAN) to back up and restore Oracle Real Application Clusters (Oracle RAC) databases. This chapter also describes Oracle RAC instance recovery, parallel backup, recovery with SQL*Plus, and using the Flash Recovery Area in Oracle RAC. The topics in this chapter include:

- RMAN Backup Scenario for Noncluster File System Backups
- RMAN Restore Scenarios for Oracle Real Application Clusters
- Instance Recovery in Oracle Real Application Clusters
- Media Recovery in Oracle Real Application Clusters
- Parallel Recovery in Oracle Real Application Clusters
- Using a Flash Recovery Area in Oracle Real Application Clusters

---

Note: For restore and recovery in Oracle RAC environments, you do not have to configure the instance that performs the recovery to also be the sole instance that restores all of the datafiles. In Oracle RAC, datafiles are accessible from every node in the **cluster**, so any node can restore archived redo log files.

See Also: the *Oracle Clusterware Administration and Deployment Guide*. for information about backing up and restoring the Oracle Clusterware components such as the Oracle Cluster Registry (OCR) and the voting disk

RMAN Backup Scenario for Noncluster File System Backups

In a non**cluster file system** environment, each node can back up only its own local archived redo log files. For example, node 1 cannot access the archived redo log files on node 2 or node 3 unless you configure the network file system for remote access. If you configure a network file system file for backups, then each node will backup its archived redo logs to a local directory.

RMAN Restore Scenarios for Oracle Real Application Clusters

This section describes the following common RMAN restore scenarios:

- Cluster File System Restore Scheme
- Noncluster File System Restore Scheme

- Using RMAN or Enterprise Manager to Restore the Server Parameter File (SPFILE)

Note: The restore and recovery procedures in a cluster file system scheme do not differ substantially from Oracle single-instance scenarios.

Cluster File System Restore Scheme

The scheme that this section describes assumes that you are using the "Automatic Storage Management and Cluster File System Archiving Scheme" on page 5-6. In this scheme, assume that node 3 performed the backups to a CFS. If node 3 is available for the restore and recovery operation, and if all of the archived logs have been backed up or are on disk, then run the following commands to perform complete recovery:

```
RESTORE DATABASE;
RECOVER DATABASE;
```

If node 3 performed the backups but is unavailable, then configure a media management device for one of the remaining nodes and make the backup media from node 3 available to this node.

Noncluster File System Restore Scheme

The scheme that this section describes assumes that you are using the "Noncluster File System Local Archiving Scheme" on page 5-7. In this scheme, each node archives locally to a different directory. For example, node 1 archives to `/arc_dest_1`, node 2 archives to `/arc_dest_2`, and node 3 archives to `/arc_dest_3`. You must configure a network file system file so that the recovery node can read the archiving directories on the remaining nodes.

If all nodes are available and if all archived redo logs have been backed up, then you can perform a complete restore and recovery by mounting the database and running the following commands from any node:

```
RESTORE DATABASE;
RECOVER DATABASE;
```

Because the network file system configuration enables each node read access to the other nodes, then the recovery node can read and apply the archived redo logs located on the local and remote disks. No manual transfer of archived redo logs is required.

Using RMAN or Enterprise Manager to Restore the Server Parameter File (SPFILE)

RMAN can restore the server parameter file either to the default location or to a location that you specify.

You can also use Enterprise Manager to restore SPFILE. From the Backup/Recovery section of the **Maintenance** tab, click **Perform Recovery**. The **Perform Recovery** link is context-sensitive and navigates you to the SPFILE restore only when the database is closed.

Instance Recovery in Oracle Real Application Clusters

Instance failure occurs when software or hardware problems disable an instance. After instance failure, Oracle automatically uses the online redo logs to perform recovery as described in this section.

Single Node Failure in Oracle Real Application Clusters

Instance recovery in Oracle RAC does not include the recovery of applications that were running on the failed instance. Oracle Clusterware restarts the instance automatically. You can also use callout programs as described in the example on Oracle Technology Network (OTN) to trigger application recovery.

Applications that were running continue by using failure recognition and recovery. This provides consistent and uninterrupted service in the event of hardware or software failures. When one instance performs recovery for another instance, the surviving instance reads online redo logs generated by the failed instance and uses that information to ensure that committed transactions are recorded in the database. Thus, data from committed transactions is not lost. The instance performing recovery rolls back transactions that were active at the time of the failure and releases resources used by those transactions.

Note: All online redo logs must be accessible for instance recovery. Therefore, Oracle recommends that you mirror your online redo logs.

Multiple-Node Failures in Oracle Real Application Clusters

When multiple node failures occur, as long as one instance survives, Oracle RAC performs instance recovery for any other instances that fail. If all instances of an Oracle RAC database fail, then Oracle automatically recovers the instances the next time one instance opens the database. The instance performing recovery can mount the database in either **cluster database** or exclusive mode from any node of an Oracle RAC database. This recovery procedure is the same for Oracle running in shared mode as it is for Oracle running in exclusive mode, except that one instance performs instance recovery for all of the failed instances.

Using RMAN to Create Backups in Oracle Real Application Clusters

Oracle provides RMAN for backing up and restoring the database. RMAN enables you to back up, restore, and recover datafiles, control files, SPFILEs, and archived redo logs. RMAN is included with the Oracle server and it is installed by default. You can run RMAN from the command line or you can use it from the Backup Manager in Oracle Enterprise Manager. In addition, RMAN is the recommended backup and recovery tool if you are using Automatic Storage Management (ASM).

The procedures for using RMAN in Oracle RAC environments do not differ substantially from those for Oracle single-instance environments. See the Oracle Backup and Recovery documentation set for more information about single-instance RMAN backup procedures.

Channel Connections to Cluster Instances

Channel connections to the instances are determined using the connect string defined by channel configurations. For example, in the following configuration, three channels are allocated using `user1/pwd1@`*`service_name`*. If you configure the SQL Net service name with load balancing turned on, then he channels are allocated at a node as decided by the load balancing algorithm.

```
CONFIGURE DEVICE TYPE sbt PARALLELISM 3;
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
CONFIGURE CHANNEL DEVICE TYPE SBT CONNECT 'user1/pwd1@service_name'
```

However, if the service name used in the connect string is not for load balancing, then you can control at which instance the channels are allocated using separate connect strings for each channel configuration as follows:

```
CONFIGURE DEVICE TYPE sbt PARALLELISM 3;
CONFIGURE CHANNEL 1.. CONNECT 'user1/pwd1@node1';
CONFIGURE CHANNEL 2.. CONNECT 'user2/pwd2@node2';
CONFIGURE CHANNEL 3.. CONNECT 'user3/pwd3@node3';
```

In the previous example, it is assumed that `node1`, `node2` and `node3` are SQL Net service names that connect to pre-defined nodes in your Oracle RAC environment. Alternatively, you can also use manually allocated channels to backup your database files. For example, the following command backs up the SPFILE, controlfile, datafiles and archived redo logs:

```
RUN
{
    ALLOCATE CHANNEL CH1 CONNECT 'user1/pwd1@node1';
    ALLOCATE CHANNEL CH2 CONNECT 'user2/pwd2@node2';
    ALLOCATE CHANNEL CH3 CONNECT 'user3/pwd3@node3';
    BACKUP DATABASE PLUS ARCHIVED LOG;
}
```

During a backup operation, as long as at least one of the channels allocated has access to the archived log, RMAN automatically schedules the backup of the specific log on that channel. Because the control file, SPFILE, and datafiles are accessible by any channel, the backup operation of these files is distributed across the allocated channels.

For a local archiving scheme, there must be at least one channel allocated to all of the nodes that write to their local archived logs. For a CFS archiving scheme, assuming that every node writes to the archived logs in the same CFS, the backup operation of the archived logs is distributed across the allocated channels.

During a backup, the instances to which the channels connect must be either all mounted or all open. For example, if the node1 instance has the database mounted while the node2 and node3 instances have the database open, then the backup fails.

Node Affinity Awareness of Fast Connections

In some cluster database configurations, some nodes of the cluster have faster access to certain datafiles than to other datafiles. RMAN automatically detects this, which is known as node affinity awareness. When deciding which channel to use to back up a particular datafile, RMAN gives preference to the nodes with faster access to the datafiles that you want to back up. For example, if you have a three-node cluster, and if node 1 has faster read/write access to datafiles 7, 8, and 9 than the other nodes, then node 1 has greater node affinity to those files than nodes 2 and 3.

> **See Also:** *Oracle Database Backup and Recovery Reference* for more information about the `CONNECT` clause of the `CONFIGURE CHANNEL` statement

Deleting Archived Redo Logs after a Successful Backup

Assuming that you have configured the automatic channels as defined in section "Channel Connections to Cluster Instances" on page 6-3, you can use the following example to delete the archived logs that you backed up *n* times. The device type can be `DISK` or `SBT`:

```
DELETE ARCHIVELOG ALL BACKED UP n TIMES TO DEVICE TYPE device_type;
```

During a delete operation, as long as at least one of the channels allocated has access to the archived log, RMAN will automatically schedule the deletion of the specific log on that channel. For a local archiving scheme, there must be at least one channel allocated that can delete an archived log. For a CFS archiving scheme, assuming that every node writes to the archived logs on the same CFS, the archived log can be deleted by any allocated channel.

If you have not configured automatic channels, then you can manually allocate the maintenance channels as follows and delete the archived logs.

```
ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE DISK CONNECT 'SYS/oracle@node1';
ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE DISK CONNECT 'SYS/oracle@node2';
ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE DISK CONNECT 'SYS/oracle@node3';
DELETE ARCHIVELOG ALL BACKED UP n TIMES TO DEVICE TYPE device_type;
```

Autolocation for Backup and Restore Commands

RMAN automatically performs autolocation of all files that it needs to back up or restore. If you use the noncluster file system local archiving scheme, then a node can only read the archived redo logs that were generated by an instance on that node. RMAN never attempts to back up archived redo logs on a channel it cannot read.

During a restore operation, RMAN automatically performs the autolocation of backups. A channel connected to a specific node only attempts to restore files that were backed up to the node. For example, assume that log sequence 1001 is backed up to the drive attached to node 1, while log 1002 is backed up to the drive attached to node 2. If you then allocate channels that connect to each node, then the channel connected to node 1 can restore log 1001 (but not 1002), and the channel connected to node 2 can restore log 1002 (but not 1001).

Media Recovery in Oracle Real Application Clusters

Media recovery must be user-initiated through a client application, whereas instance recovery is automatically performed by the database. In these situations, use RMAN to restore backups of the datafiles and then recover the database. The procedures for RMAN media recovery in Oracle RAC environments do not differ substantially from the media recovery procedures for single-instance environments.

The node that performs the recovery must be able to restore all of the required datafiles. That node must also be able to either read all of the required archived redo logs on disk or be able to restore them from backups.

When recovering a database with encrypted tablespaces (for example after a `SHUTDOWN ABORT` or a catastrophic error that brings down the database instance), you must open the Oracle Wallet after database mount and before you open the database, so the recovery process can decrypt data blocks and redo.

Parallel Recovery in Oracle Real Application Clusters

Oracle automatically selects the optimum degree of parallelism for instance, crash, and media recovery. Oracle applies archived redo logs using an optimal number of parallel processes based on the availability of CPUs. You can use parallel instance recovery and parallel media recovery in Oracle RAC databases as described under the following topics:

- Parallel Recovery with RMAN

- Disabling Parallel Recovery

See Also: *Oracle Database Backup and Recovery User's Guide* for more information on these topics

Parallel Recovery with RMAN

With RMAN's `RESTORE` and `RECOVER` commands, Oracle automatically makes parallel the following three stages of recovery:

Restoring Datafiles When restoring datafiles, the number of channels you allocate in the RMAN recover script effectively sets the parallelism that RMAN uses. For example, if you allocate five channels, you can have up to five parallel streams restoring datafiles.

Applying Incremental Backups Similarly, when you are applying incremental backups, the number of channels you allocate determines the potential parallelism.

Applying Archived Redo Logs With RMAN, the application of archived redo logs is performed in parallel. Oracle automatically selects the optimum degree of parallelism based on available CPU resources.

Disabling Parallel Recovery

You can override this using the procedures under the following topics:

- Disabling Instance and Crash Recovery Parallelism
- Disabling Media Recovery Parallelism

Disabling Instance and Crash Recovery Parallelism

To disable parallel instance and crash recovery on a system with multiple CPUs, set the `RECOVERY_PARALLELISM` parameter to 0.

Disabling Media Recovery Parallelism

Use the `NOPARALLEL` clause of the RMAN `RECOVER` command or the `ALTER DATABASE RECOVER` statement to force Oracle to use non-parallel media recovery.

Using a Flash Recovery Area in Oracle Real Application Clusters

To use a flash recovery area in Oracle RAC, you must place it on an ASM disk group, a Cluster File System, or on a shared directory that is configured through a network file system file for each Oracle RAC instance. In other words, the flash recovery area must be shared among all of the instances of an Oracle RAC database. In addition, set the parameter `DB_RECOVERY_FILE_DEST` to the same value on all instances.

Enterprise Manager enables you to set up a flash recovery area. To use this feature:

1. From the Cluster Database home page, click the **Maintenance** tab.
2. Under the Backup/Recovery options list, click **Configure Recovery Settings**.
3. Specify your requirements in the Flash Recovery Area section of the page.
4. Click **Help** on this page for more information.

7

Using Cloning to Add ASM and Oracle RAC to Nodes in a New Cluster

This chapter describes how to clone Oracle Automatic Storage Management (ASM) and Oracle Real Application Clusters (Oracle RAC) database homes on Linux and UNIX systems to nodes in a new cluster. To extend Oracle RAC to nodes in an existing cluster, see Chapter 8.

This chapter describes a noninteractive cloning technique that you implement through the use of scripts. The cloning techniques described in this chapter are best suited for performing multiple simultaneous cluster installations. Creating the scripts is a manual process and can be error prone. If you only have one cluster to install, then you should use the traditional automated and interactive installation methods, such as Oracle Universal Installer (OUI), or the Provisioning Pack feature of Oracle Enterprise Manager.

Note: Cloning is not a replacement for Oracle Enterprise Manager cloning that is a part of the Provisioning Pack. During Enterprise Manager cloning, the provisioning process interactively asks you the details about the Oracle home (such as the location to which you want to deploy the clone, the name of the Oracle Database home, a list of the nodes in the cluster, and so on).

The Provisioning Pack feature of Oracle Grid Control provides a framework to make it easy for you to automate the provisioning of new nodes and clusters. For data centers with many Oracle RAC clusters, the investment in creating a cloning procedure to easily provision new clusters and new nodes to existing clusters is worth the effort.

This chapter contains the following topics:

- Introduction to Cloning Oracle ASM and Oracle RAC
- Preparing to Clone ASM and Oracle RAC
- Deploying ASM and Oracle RAC to Other Nodes in the Cluster
- Locating and Viewing Log Files Generated During Cloning

Introduction to Cloning Oracle ASM and Oracle RAC

Cloning is the process of copying an existing Oracle installation to a different location and updating the copied bits to work in the new environment. The changes made by

one-off patches applied on the source Oracle home, would also be present after the clone operation. The source and the destination path (host to be cloned) need not be the same.

Some situations in which cloning is useful are:

- Cloning provides a way to prepare an ASM and an Oracle RAC home once and deploy it to many hosts simultaneously. You can complete the installation silently, as a noninteractive process. You do not need to use a graphical user interface (GUI) console and you can perform cloning from an Secure Shell (SSH) terminal session, if required.
- Cloning enables you to create a new installation (copy of a production, test, or development installation) with all patches applied to it in a single step. Once you have performed the base installation and applied all patchsets and patches on the source system, the clone performs all of these individual steps as a single procedure. This is in contrast to going through the installation process to perform the separate steps to install, configure, and patch the installation on each node in the cluster.
- Installing ASM and Oracle RAC by cloning is a very quick process. For example, cloning an Oracle RAC home to a new cluster of more than two nodes requires a few minutes to install the Oracle base software, plus a few minutes more for each node (approximately the amount of time it takes to run the `root.sh` script).
- Cloning provides a guaranteed method of repeating the same Oracle installation on multiple clusters.

The cloned installation behaves the same as the source installation. For example, the cloned Oracle home can be removed using OUI or patched using OPatch. You can also use the cloned Oracle home as the source for another cloning operation. You can create a cloned copy of a test, development, or production installation by using the command-line cloning scripts. The default cloning procedure is adequate for most usage cases. However, you can also customize various aspects of cloning, for example, to specify custom port assignments, or to preserve custom settings.

The cloning process works by copying all of the files from the source Oracle home to the destination Oracle home. Thus, any files used by the source instance that are located outside the source Oracle home's directory structure are not copied to the destination location.

The size of the binaries at the source and the destination may differ because these are relinked as part of the clone operation and the operating system patch levels may also differ between these two locations. Additionally, the number of files in the cloned home would increase because several files copied from the source, specifically those being instantiated, are backed up as part of the clone operation.

Preparing to Clone ASM and Oracle RAC

In the preparation phase, you create a copy of an Oracle Database home that you then use to perform the cloning procedure on one or more nodes, and you install Oracle Clusterware.

Step 1 Install the Oracle Database software

Using the detailed instructions in your platform-specific Oracle Real Application Clusters installation guide to install the Oracle Database software and patches:

1. Install Oracle Database Release 11*g* and choose the **Software only** installation option.

2. Patch the release to the required level (for example, 11.1.0.*n*).
3. Apply one-off patches, if necessary.

Step 2 Create a backup of the source home

Create a copy of the Oracle Database home. You will use this file to copy the Oracle Database home to each node in the cluster (as described in the "Deploying ASM and Oracle RAC to Other Nodes in the Cluster" section on page 7-3).

When creating the backup (tar) file, the best practice is to include the release number in the name of the file. For example:

```
# cd /opt/oracle/product/11g/db_1
# tar -zcvf /pathname/db11101.tgz .
```

Step 3 Install and start Oracle Clusterware

Before you can use cloning to create a new Oracle RAC home, Oracle Clusterware must be installed and started on the new nodes. In other words, you extend the software onto the new nodes in the same order that you installed the Oracle Clusterware and Oracle database software components on the original nodes.

> **See Also:** *Oracle Clusterware Administration and Deployment Guide* for information about cloning Oracle Clusterware homes to create new clusters, and starting Oracle Clusterware by issuing the `crsctl start crs` command

Deploying ASM and Oracle RAC to Other Nodes in the Cluster

After you complete the prerequisite tasks described in the "Preparing to Clone ASM and Oracle RAC" section on page 7-2, you can deploy new ASM and Oracle RAC homes. The deployment steps in this section follow the best practices, which recommend deploying two Oracle Database homes on each node: one home is for the ASM instance and the other home is for the Oracle RAC database instance.

The following sections provide step-by-step instructions for:

- Deploying ASM Instance Homes
- Deploying Oracle RAC Database Homes

You can script the multiple-step processes described in these sections to run automatically, as a silent installation.

Deploying ASM Instance Homes

You must deploy the ASM home before you can deploy the new Oracle RAC database home. This section provides step-by-step instructions that describe how to:

1. Prepare the new cluster nodes
2. Deploy the ASM software
3. Run the clone.pl script on each node
4. Run the ASM_home/root.sh script on each node
5. Run NETCA to create the listeners
6. Run DBCA to create the ASM instances on each node

See Also: *Oracle Database Storage Administrator's Guide* for complete information about ASM

Step 1 Prepare the new cluster nodes

Perform the Oracle Database preinstallation steps, including such things as:

- Specify the kernel parameters.
- Use short, nondomain-qualified names for all names in the Hosts file.
- Test whether or not the interconnect interfaces are reachable using the `ping` command.

See your platform-specific Oracle RAC installation guide for a complete preinstallation checklist.

Note: Unlike traditional methods of installation, the cloning process does not validate your input during the preparation phase. (By comparison, during the traditional method of installation using the OUI, various checks take place during the interview phase.) Thus, if you make any mistakes during the hardware setup or in the preparation phase, then the cloned installation will fail.

Step 2 Deploy the ASM software

To deploy the ASM software, you need to:

1. Restore the ASM home to all nodes in either the original home directory or to a different directory path. For example:

   ```
   [root@node1 root]# mkdir -p /opt/oracle/product/11g/asm
   [root@node1 root]# cd /opt/oracle/product/11g/asm
   [root@node1 asm]# tar -zxvf /pathname/asm11101.tgz
   ```

   Note that the ASM home location does not have to be in the same directory path as the original source home directory that you used to create the tar file.

2. Change the ownership of all files to the `oracle` and `oinstall` group. For example:

   ```
   [root@node1 asm]# chown -R oracle:oinstall /opt/oracle/product/11g/asm
   ```

Note: You can perform this step at the same time you perform steps 3 and 4 to run the `clone.pl` and *`ASM_home`*`/root.sh` scripts on each cluster node.

Step 3 Run the clone.pl script on each node

To run the `clone.pl` script, which performs the main ASM cloning tasks, you must:

- Supply environment variables and cloning parameters in a `start.sh` script, as described in Table 7–1 and Table 7–2. Because the `clone.pl` script is sensitive to the parameters being passed to it, you must be accurate in your use of brackets, single quotes, and double quotes.
- Invoke the script as the `oracle` operating system user.

Example 7–1 shows an excerpt from the `start.sh` script that calls the `clone.pl` script.

Example 7–1 Excerpt From the start.sh Script to Clone ASM

```
ORACLE_BASE=/opt/oracle
ASM_home=/opt/oracle/product/11g/asm
cd ASM_home/clone
THISNODE=`hostname -s`
E01=ORACLE_HOME=${ASM_home}
E02=ORACLE_HOME_NAME=OraDBASM
E03=ORACLE_BASE=/opt/oracle
C01="-O'\"CLUSTER_NODES={node1, node2}\"'"
C02="'-O\"LOCAL_NODE=$THISNODE\"'"
perl ASM_home/clone/bin/clone.pl $E01 $E02 $E03 $C01 $C02
```

See Also: Example 7–1, "Excerpt From the start.sh Script to Clone ASM"

Table 7–1 describes the environment variables E01, E02, and E03 that are shown in bold typeface in Example 7–1.

Table 7–1 Environment Variables Passed to the clone.pl Script

Symbol	Variable	Description
E01	ORACLE_HOME	The location of the ASM home. This directory location must exist and must be owned by the oracle operating system group: oinstall.
E02	ORACLE_HOME_NAME	The name of the Oracle home for the ASM home. This is stored in the Oracle Inventory.
E03	ORACLE_BASE	The location of the Oracle Base directory.

Table 7–2 describes the cloning parameters C01 and C02, that are shown in bold typeface in Example 7–1.

Table 7–2 Cloning Parameters Passed to the clone.pl Script.

Variable	Name	Parameter	Description
C01	Cluster Nodes	CLUSTER_NODES	Lists the nodes in the cluster.
C02	Local Node	LOCAL_NODE	The name of the local node.

Step 4 Run the ASM_home/root.sh script on each node

Run the *ASM_home*/root.sh as the root operating system user as soon as the clone.pl procedure completes on the node.

You should set the LD_LIBRARY_PATH environment variable before running the root.sh script. For example:

```
[root@node1 root]# export LD_LIBRARY_PATH=ASM_home/lib:$LD_LIBRARY_PATH
[root@node1 root]# /opt/oracle/product/11g/asm/root.sh -silent
```

Note that you can run the script on each node simultaneously:

```
[root@node2 root]# export LD_LIBRARY_PATH=ASM_home/lib:$LD_LIBRARY_PATH
[root@node2 root]# /opt/oracle/product/11g/asm/root.sh -silent
```

Ensure the script has completed on each node before proceeding to the next step.

Step 5 Run NETCA to create the listeners

At the end of the Oracle ASM home installation, the OUI creates a listener on each node, and registers the nodes as CRS resources with Oracle Clusterware.

The following example shows how to run NETCA in silent mode to create the listeners. In the response file, provide your node names in place of the variables: *node1*,*node2*. NETCA uses the response file to create the `listener.ora` entries on each node, start the listeners, and add the listeners to the Oracle Cluster Registry (OCR).

```
[oracle@node1 oracle]$ cd $ORACLE_HOME/bin/
[oracle@node1 bin]$ ./netca /silent \
                    /responseFile $ORACLE_HOME/network/install/netca_typ.rsp \
                    /inscomp server \
                    /nodeinfo node1,node2
```

Step 6 Run DBCA to create the ASM instances on each node

This step shows how to invoke the DBCA in silent mode and provide response file input to create the ASM instances and the first ASM disk group.

The following example creates ASM instances on each node, registers them in the Oracle Cluster Registry, and creates an ASM disk group called `+DATA` that is made up of four disks: `sde1`, `sdf1`,`sdg1` and `sdh1`. It also sets the SYS password to `mypassword`:

```
[oracle@node1 oracle]$ export ORACLE_HOME=/opt/oracle/product/11g/asm
[oracle@node1 oracle]$ cd $ORACLE_HOME/bin/
[oracle@node1 bin]$ ./dbca -silent -configureASM -gdbName NO -sid NO \
                     -emConfiguration NONE \
                     -diskList "/dev/sde1,/dev/sdf1,/dev/sdg1,/dev/sdh1" \
                     -diskString "/dev/sd[e-h]1" \
                     -diskGroupName "DATA" \
                     -datafileJarLocation $ORACLE_HOME/assistants/dbca/templates \
                     -nodeinfo node1,node2 \
                     -obfuscatedPasswords false \
                     -oratabLocation /etc/oratab \
                     -asmSysPassword mypassword \
                     -redundancy EXTERNAL
```

See Also: *Oracle Database 2 Day DBA* for information about using DBCA to create and configure a database

Deploying Oracle RAC Database Homes

Deployment to a new cluster of the Oracle RAC Database home is a multiple-step process. The steps are almost identical to the steps you performed to for creating the ASM home described in "Deploying ASM Instance Homes" on page 7-3. The differences are that when deploying an Oracle RAC database, you do not create listeners inside the database home and you provide different DBCA parameters to deploy an Oracle RAC database home.

This section provides step-by-step instructions that describe how to:

1. Prepare the new cluster nodes
2. Deploy the Oracle RAC database software
3. Run the clone.pl script on each node
4. Run the $ORACLE_HOME/root.sh script on each node

5. Run DBCA to create the Oracle RAC instances on each node

Step 1 Prepare the new cluster nodes

Perform the Oracle Database preinstallation steps, including such things as:

- Specify the kernel parameters.
- Use short, nondomain-qualified names for all names in the Hosts file.
- Verify that you can ping the public and interconnect names.
- Ensure Oracle Clusterware is active.
- Ensure ASM is active and there is at least one ASM disk group configured.

See your platform-specific Oracle RAC installation guide for a complete preinstallation checklist.

Step 2 Deploy the Oracle RAC database software

To deploy the Oracle RAC database software, you need to:

1. Restore the Oracle Database home to all nodes. For example:

```
[root@node1 root]# mkdir -p /opt/oracle/product/11g/db
[root@node1 root]# cd /opt/oracle/product/11g/db
[root@node1 db]# tar -zxvf /pathname/db11101.tgz
```

   When providing the home location and *pathname*:

   - If you are cloning Oracle RAC to a new cluster, then the home location can be in the same directory path or in a different directory path from the source home that you used to create the tar.
   - If you are cloning Oracle RAC to an existing cluster, then the home location and directory paths must be the same.

2. Change the ownership of all files to the `oracle` and `oinstall` group. For example:

```
[root@node1 db]# chown -R oracle:oinstall /opt/oracle/product/11g/db
```

> **Note:** You can perform this step at the same time you perform steps 3 and 4 to run the `clone.pl` and `ORACLE_HOME/root.sh` scripts on each cluster node.

Step 3 Run the clone.pl script on each node

To run the `clone.pl` script, which performs the main Oracle RAC cloning tasks, you must:

- Supply the environment variables and cloning parameters in the start.sh script, as described in Table 7–3 and Table 7–4. Because the `clone.pl` script is sensitive to the parameters being passed to it, you must be accurate in your use of brackets, single quotes, and double quotes.
- Invoke the script as the `oracle` operating system user.

Example 7–2 shows an excerpt from the `start.sh` script that calls the `clone.pl` script.

Example 7–2 Excerpt From the start.sh Script to Clone the Oracle RAC Database

```
ORACLE_BASE=/opt/oracle
ORACLE_HOME=/opt/oracle/product/11g/db
cd $ORACLE_HOME/clone
THISNODE=`hostname -s`

E01=ORACLE_HOME=${ORACLE_HOME}
E02=ORACLE_HOME_NAME=OraDBRAC
E03=ORACLE_BASE=/opt/oracle
C01="-O'\"CLUSTER_NODES={node1, node2}\"'"
C02="'-O\"LOCAL_NODE=$THISNODE\"'"

perl $ORACLE_HOME/clone/bin/clone.pl $E01 $E02 $E03 $C01 $C02
```

Table 7–3 describes the environment variables E01, E02, and E03 that are shown in bold typeface in Example 7–2.

Table 7–3 Environment Variables Passed to the clone.pl Script

Symbol	Variable	Description
E01	ORACLE_HOME	The location of the Oracle RAC database home. This directory location must exist and must be owned by the oracle operating system group: oinstall.
E02	ORACLE_HOME_NAME	The name of the Oracle home for the Oracle RAC database. This is stored in the Oracle Inventory.
E03	ORACLE_BASE	The location of the Oracle Base directory.

Table 7–4 describes the cloning parameters C01 and C02, that are shown in bold typeface in Example 7–2.

Table 7–4 Cloning Parameters Passed to the clone.pl Script.

Variable	Name	Parameter	Description
C01	Cluster Nodes	CLUSTER_NODES	Lists the nodes in the cluster.
C02	Local Node	LOCAL_NODE	The name of the local node.

Step 4 Run the $ORACLE_HOME/root.sh script on each node

Run the $ORACLE_HOME/root.sh as the root operating system user as soon as the clone.pl procedure completes on the node.

You should set the LD_LIBRARY_PATH environment variable before running the root.sh script. For example:

```
[root@node1 root]# export LD_LIBRARY_PATH=$ORACLE_HOME/lib:$LD_LIBRARY_PATH
[root@node1 root]# /opt/oracle/product/11g/db/root.sh -silent
```

Note that you can run the script on each node simultaneously:

```
[root@node2 root]# export LD_LIBRARY_PATH=$ORACLE_HOME/lib:$LD_LIBRARY_PATH
[root@node2 root]# /opt/oracle/product/11g/db/root.sh -silent
```

Ensure the script has completed on each node before proceeding to the next step.

Step 5 Run DBCA to create the Oracle RAC instances on each node

This step shows how to invoke the DBCA in silent mode and provide response file input to create the Oracle RAC instances.

The following example creates Oracle RAC instances on each node, registers the instances in the OCR, creates the database files in the ASM disk group called DATA, and creates sample schemas. It also sets the sys, system, sysman and dbsnmp passwords to mypassword:

```
[oracle@node1 oracle]$ export ORACLE_HOME=/opt/oracle/product/11g/db
[oracle@node1 oracle]$ cd $ORACLE_HOME/bin/
[oracle@node1 bin]$./dbca -silent -createDatabase -templateName General_Purpose.dbc\
                  -gdbName ERI -sid ERI \
                  -sysPassword mypassword -systemPassword mypassword \
                  -sysmanPassword mypassword -dbsnmpPassword mypassword \
                  -emConfiguration LOCAL \
                  -storageType ASM -diskGroupName DATA \
                  -datafileJarLocation $ORACLE_HOME/assistants/dbca/templates \
                  -nodeinfo node1,node2 -characterset WE8ISO8859P1 \
                  -obfuscatedPasswords false -sampleSchema true \
                  -oratabLocation /etc/oratab
```

See Also: *Oracle Database 2 Day DBA* for information about using DBCA to create and configure a database

Locating and Viewing Log Files Generated During Cloning

The cloning script runs multiple tools, each of which may generate its own log files. After the clone.pl script finishes running, you can view log files to obtain more information about the cloning process.

The following log files that are generated during cloning are the key log files of interest for diagnostic purposes:

- *Central_Inventory*/logs/cloneActions timestamp.log

  Contains a detailed log of the actions that occur during the OUI part of the cloning.

- *Central_Inventory*/logs/oraInstall timestamp.err

  Contains information about errors that occur when OUI is running.

- *Central_Inventory/logs/oraInstall timestamp.out*

  Contains other miscellaneous messages generated by OUI.

- $ORACLE_HOME/clone/logs/clone timestamp.log

  Contains a detailed log of the actions that occur prior to cloning as well as during the cloning operations.

- $ORACLE_HOME/clone/logs/error timestamp.log

  Contains information about errors that occur prior to cloning as well as during cloning operations.

Table 7–5 describes how to find the location of the Oracle inventory directory.

Table 7–5 Finding the Location of the Oracle Inventory Directory

Type of System ,,,	Location of the Oracle Inventory Directory
All UNIX computers except Linux and IBM AIX	/var/opt/oracle/oraInst.loc
IBM AIX and Linux	/etc/oraInst.loc file.

Table 7–5 (Cont.) Finding the Location of the Oracle Inventory Directory

Type of System ,,,	Location of the Oracle Inventory Directory
Windows	Obtain the location from the Windows Registry key: `HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\INST_LOC`

8

Using Cloning to Extend Oracle RAC to Nodes in the Same Cluster

This chapter provides information about using cloning to extend Oracle Real Application Clusters (Oracle RAC) to additional nodes in an existing cluster. To add ASM and Oracle RAC to nodes in a new cluster, see Chapter 7.

> **See Also:** "Introduction to Cloning Oracle ASM and Oracle RAC" on page 7-1 for an overview of cloning and a discussion about the benefits of cloning

Adding Nodes Using Cloning in Oracle Real Application Clusters Environments

This section explains how to add nodes to existing Oracle RAC environments by using Oracle cloning for Linux and UNIX system environments.

The cloning procedures assume that you have successfully installed and configured an Oracle RAC environment to which you want to add nodes and instances. To add nodes to an Oracle RAC environment using cloning, first extend the Oracle Clusterware configuration, then extend the Oracle Database software with Oracle RAC, and then add the listeners and instances by running the Oracle assistants

Complete the following steps to clone Oracle Database with Oracle RAC software on Linux and UNIX systems:

1. Follow the steps in the "Preparing to Clone ASM and Oracle RAC" on page 7-2 to create a copy of an Oracle home that you then use to perform the cloning procedure on one or more nodes.
2. If you do not have a shared Oracle Database home, then tar the Oracle home from the existing node and copy it to the new node. Assume that the location of the destination Oracle home on the new node is `$ORACLE_HOME`. Otherwise, skip this step.
3. If you do not have a shared Oracle Database home, then on the new node go to the `$ORACLE_HOME/clone/bin` directory and run the following command where *`existing_node`* is the name of the node that you are cloning, *`new_node2`* and *`new_node3`* are the names of the new nodes, and *`Oracle_home_name`* is the name of the Oracle home:

```
perl clone.pl '-O"CLUSTER_NODES={existing_node,new_node2,new_node3}"'
 '-O"LOCAL_NODE=new_node2"' ORACLE_BASE=/opt/oracle ORACLE_HOME=$ORACLE_HOME
 ORACLE_HOME_NAME=Oracle_home_name '-O-noConfig'
```

If you have a shared Oracle Database home, then append the `-cfs` option to the command example in this step for the cluster file system:

```
perl clone.pl '-O"CLUSTER_NODES={existing_node,new_node2,new_node3}"'
'-O"LOCAL_NODE=new_node2"' ORACLE_BASE=/opt/oracle ORACLE_HOME=$ORACLE_HOME
 ORACLE_HOME_NAME=Oracle_home_name '-O-noConfig'  '-O-cfs'
```

4. Run the following command on the existing node from the `$ORACLE_HOME/oui/bin` directory where *`existing_node`* is the name of the original node that you are cloning and *`new_node2`* and *`new_node3`* are the names of the new node:

   ```
   ./runInstaller -updateNodeList ORACLE_HOME=$ORACLE_HOME "CLUSTER_
   NODES={existing_node,new_node2,new_node3}"
   ```

5. On the new node, go to the `$ORACLE_HOME` directory and run the following command:

   ```
   ./root.sh
   ```

6. On the new node, run Net Configuration Assistant (NETCA) to add a listener.
7. From the node that you cloned, run Database Configuration Assistant (DBCA) to add the new instance.

The cloning script runs multiple tools, each of which may generate its own log files. After the `clone.pl` script finishes running, you can view log files to obtain more information about the cloning process. See "Locating and Viewing Log Files Generated During Cloning" on page 7-9 for more information.

9

Adding and Deleting Oracle RAC from Nodes on Linux and UNIX Systems

This chapter describes how to use the `addNode.sh` and `rootdelete.sh` scripts to extend an existing Oracle Real Application Clusters (Oracle RAC) home to other nodes and instances in the cluster, and delete Oracle RAC from nodes and instances in the cluster. This chapter provides instructions for Linux and UNIX systems.

If your goal is to clone an existing Oracle RAC home to create multiple new Oracle RAC installations across the cluster, then use the cloning procedures that are described in Chapter 7, "Using Cloning to Add ASM and Oracle RAC to Nodes in a New Cluster".

The topics in this chapter include the following:

- Adding Oracle RAC to Nodes Running Clusterware and Oracle Database
- Adding Oracle RAC to Nodes That Do Not Have Clusterware and Oracle Database
- Deleting Cluster Nodes from Oracle Real Application Clusters Environments

---

Note: The phrase "target node" as used in this chapter refers to the node to which you plan to extend Oracle RAC environment.

See Also: *Oracle Database 2 Day + Real Application Clusters Guide* for additional information about configuring a new Oracle RAC cluster or scaling up an existing Oracle RAC cluster

Adding Oracle RAC to Nodes Running Clusterware and Oracle Database

Before beginning this procedure, ensure that your existing nodes have the correct path to the *`CRS_home`* and that the `$ORACLE_HOME` environment variables is set correctly.

To add Oracle RAC to nodes that already have Oracle Clusterware and Oracle Database software installed, you must configure the target nodes with the Oracle Database software that is on the existing nodes of the cluster. To do this, perform the following steps to run two versions of an OUI process: one for the Oracle Clusterware and one for the database layer:

1. Add Oracle RAC to target nodes at the Oracle Clusterware layer by running OUI from the Oracle Clusterware home on an existing node, using the following command:

   ```
   CRS_home/oui/bin/addNode.sh -noCopy
   ```

2. Add Oracle RAC to target nodes at the Oracle software layer by running OUI from the Oracle home as follows:

```
Oracle_home/oui/bin/addNode.sh -noCopy
```

In the `-noCopy` mode, OUI performs all add node operations except for the copying of software to the target nodes.

Note: Oracle recommends that you back up your voting disk and Oracle Cluster Registry (OCR) files after you complete the node addition process.

See Also: "Extending ASM to Nodes Running Single-Instance or Oracle RAC Databases" on page 2-6 for complete information about adding ASM instances to new nodes

Adding Oracle RAC to Nodes That Do Not Have Clusterware and Oracle Database

This section explains how to add nodes to clusters using detailed manual procedures. If the nodes to which you want to add Oracle RAC do not have clusterware or Oracle software installed on them, then complete the following steps to add Oracle RAC to the target nodes. The procedures in these steps assume that you already have an operative Linux or UNIX environment.

Otherwise, to add Oracle RAC to a node that is already configured with clusterware and Oracle software, follow the procedure described in "Adding Oracle RAC to Nodes Running Clusterware and Oracle Database" on page 9-1.

This section contains the following topics:

- Prerequisite Steps for Extending Oracle RAC to Target Nodes
- Extend Oracle Clusterware to Target Nodes
- Configure Shared Storage on Target Nodes
- Add the Oracle Real Application Clusters Database Homes to Target Nodes
- Add ASM and Oracle RAC Database Instances to Target Nodes

See Also: "Extending ASM to Nodes Running Single-Instance or Oracle RAC Databases" on page 2-6 for complete information about adding ASM instances to new nodes

Prerequisite Steps for Extending Oracle RAC to Target Nodes

The following steps describe how to set up target nodes to be part of your **cluster**:

Step 1 Make physical connections

Connect the target nodes' hardware to the network infrastructure of your cluster. This includes establishing electrical connections, configuring network interconnects, configuring shared disk subsystem connections, and so on. See your hardware vendor documentation for details about this step.

Step 2 Install the operating system

Install a cloned image of the operating system that matches the operating system on the other nodes in your cluster. This includes installing required service patches and drivers. See your hardware vendor documentation for details about this process.

> **See Also:** Your platform-specific Oracle Real Application Clusters installation guide for procedures about using the Database Configuration Assistant (DBCA) to create and delete Oracle RAC databases

Step 3 Create Oracle users

Create Oracle users.

As `root` user, create the Oracle users and groups using the same user ID and group ID as on the existing nodes.

Step 4 Verify the installation

Verify the installation with the Cluster Verification Utility (CVU) by performing the following steps:

1. From the `/bin` directory in the *`CRS_home`* on the existing nodes, run the CVU command to verify your installation at the post hardware installation stage as shown in the following example, where *`node_list`* is a comma-delimited list of nodes you want in your cluster:

   ```
   cluvfy stage -post hwos -n node_list|all [-verbose]
   ```

   > **Note:** You can only use the `all` option with the `-n` argument if you have set the `CV_NODELIST` variable to represent the list of nodes on which you want to perform the CVU operation.

   You can also use this CVU command to:

   - Verify the node is reachable, for example, to all of the nodes from the local node.
   - Verify user equivalence to all given nodes the local node, node connectivity among all of the given nodes, accessibility to shared storage from all of the given nodes, and so on.

   > **See Also:** "Using the Cluster Verification Utility" section in the *Oracle Clusterware Administration and Deployment Guide*

2. From the `/bin` directory in the *`CRS_home`* on the existing nodes, run the CVU command to obtain a detailed comparison of the properties of the reference node with all of the other nodes that are part of your current cluster environment where *`ref_node`* is a node in your existing cluster against which you want CVU to compare, for example, the target nodes that you specify with the comma-delimited list in *`node_list`* for the `-n` option, *`orainventory_group`* is the name of the Oracle inventory group, and *`osdba_group`* is the name of the OSDBA group:

   ```
   cluvfy comp peer [ -refnode ref_node ] -n node_list
   [ -orainv orainventory_group ] [ -osdba osdba_group ] [-verbose]
   ```

For the reference node, select a node from your existing cluster nodes against which you want CVU to compare, for example, the target nodes that you specify with the `-n` option.

Note: For all of the add node and delete node procedures for Linux and UNIX systems, temporary directories such as `/tmp`, `$TEMP`, or `$TMP`, *should not be* shared directories. If your temporary directories are shared, then set your temporary environment variable, such as `$TEMP`, to a nonshared location on a local node. In addition, use a directory that exists on all of the nodes.

Step 5 Check the installation

To verify that your installation is configured correctly, perform the following steps:

1. Ensure that the target nodes can access the private interconnect. This interconnect must be properly configured before you can complete the procedures described in this chapter.
2. If you are not using a cluster file system, then determine the location on which your cluster software was installed on the existing node. Make sure that you have at least 250 MB of free space on the same location on each of the target nodes to install Oracle Clusterware. In addition, ensure you have enough free space on each target node to install the Oracle binaries.
3. Ensure that the OCR and the voting disk are accessible by the target nodes using the same path. In addition, the OCR and voting disk devices must have the same permissions as on the existing nodes.
4. Verify user equivalence to and from an existing node to the target nodes using `rsh` or `SSH` on Linux and UNIX systems, or on Window systems make sure that you can run the following command from *all* of the existing nodes of your cluster where the `hostname` is the public network name of the target node:

   ```
   NET USE \\hostname\C$
   ```

   You have the required administrative privileges on each node if the operating system responds with:

   ```
   Command completed successfully.
   ```

After completing the procedures in this section, your target nodes are connected to the cluster and configured with the required software to make them visible to Oracle Clusterware.

Note: Do not change a hostname after the Oracle Clusterware installation. This includes adding or deleting a domain qualification.

Extend Oracle Clusterware to Target Nodes

Extend an existing Oracle Clusterware home to the target nodes following the instructions in *Oracle Clusterware Administration and Deployment Guide*.

If you are using Oracle Clusterware without vendor clusterware, then you can add and delete Oracle Clusterware on nodes without stopping the existing nodes. If you are using Oracle Clusterware with vendor clusterware, then you can add nodes on some Linux and UNIX systems without stopping the existing nodes if your

clusterware supports this. See your vendor-specific clusterware documentation for more information.

> **Note:** For systems using shared storage for the clusterware home, ensure that the existing clusterware is accessible by the target nodes. Also ensure that the target nodes can be brought online as part of the existing cluster.

Configure Shared Storage on Target Nodes

Use the information in this section to configure shared storage so that the target nodes can access the Oracle software, and so that the existing nodes can access the target nodes and instances. Then, use the procedure described in "Add the Oracle Real Application Clusters Database Homes to Target Nodes" on page 9-6.

> **Note:** In some cases, your current configuration may not be compatible with an ASM activity that you are trying to perform, either explicitly or with an automatic ASM extension to other nodes. If you are using DBCA to build a database using a new Oracle home, and if the ASM version is from an earlier release of the Oracle software but does not exist on all of the nodes you selected for the database, then ASM cannot be extended. Instead, the DBCA session displays an error, prompting you either to run the add node script or to upgrade ASM using the DBUA.

To extend an existing Oracle RAC database to the target nodes, configure the same type of storage on the target nodes as you are using on the existing nodes in the Oracle RAC environment:

- Automatic Storage Management (ASM)

  You do not need to install ASM on the added node because the ASM instance is created implicitly upon node addition. If you are using ASM for storage, then make sure that the target nodes can access the ASM disks with the same permissions as the existing nodes.

  > **See Also:** "Extending ASM to Nodes Running Single-Instance or Oracle RAC Databases" on page 2-6 for instructions about adding a new ASM instance to a node running either a single-instance database or an Oracle RAC database instance in a cluster

- Oracle Cluster File System (OCFS)

  If you are using Oracle Cluster File Systems, then make sure that the target nodes can access the cluster file systems in the same way that the other nodes access them.

  Run the following command to verify your cluster file system and obtain detailed output where *nodelist* includes both the preexisting nodes and the target nodes and *file system* is the name of the file system that you used for the Oracle Cluster File System:

  ```
  cluvfy comp cfs -n nodelist -f file system [-verbose]
  ```

> **See Also:** *Oracle Clusterware Administration and Deployment Guide* for more information about enabling and using the CVU, and your platform-specific Oracle Clusterware installation guide for more information about the Oracle Cluster File System

- Vendor cluster file systems

  If your cluster database uses vendor cluster file systems, then configure the target nodes to use the vendor cluster file systems. See the vendor clusterware documentation for the preinstallation steps for your Linux or UNIX platform.

- Raw device storage

  If your cluster database uses raw devices, then prepare the raw devices on the target nodes, as follows:

  To prepare raw device storage on the target nodes, you need at least two new disk partitions to accommodate the redo logs for each new instance. Make these disk partitions the same size as the redo log partitions that you configured for the existing nodes' instances. Also create an additional logical partition for the undo tablespace for automatic undo management.

  On applicable operating systems, you can create symbolic links to your raw devices. Optionally, you can create a raw device mapping file and set the `DBCA_RAW_CONFIG` environment variable so that it points to the raw device mapping file. Use your vendor-supplied tools to configure the required raw storage.

  > **See Also:** Your platform-specific Oracle Real Application Clusters installation guide for procedures about using DBCA to create and delete Oracle RAC databases

  Run the following command to verify that the prepared raw device storage is accessible from all of the configured cluster nodes where *`node_list`* includes both the pre-existing nodes and the newly added nodes and *`storageID_list`* is a comma-delimited list of storage identifiers:

  ```
  cluvfy comp ssa [ -n node_list ] [ -s storageID_list ] [-verbose]
  ```

  > **See Also:** *Oracle Clusterware Administration and Deployment Guide* for more information about enabling and using the CVU

Add the Oracle Real Application Clusters Database Homes to Target Nodes

You can add the Oracle RAC database home to target nodes using either of the following methods:

- Extending the Database Home to Target Nodes Using OUI in Interactive Mode
- Extending the Database Home to Target Nodes Using OUI in Silent Mode

> **See Also:** *Oracle Universal Installer and OPatch User's Guide* for more information about how to configure command-line response files and *Oracle Database Net Services Administrator's Guide* for more information about NETCA

Extending the Database Home to Target Nodes Using OUI in Interactive Mode

To extend Oracle RAC to the target nodes, run OUI in `add node` mode to configure the Oracle home on the target nodes. If you have multiple Oracle homes, then perform the following steps for each Oracle home that you want to include on the target nodes:

1. Ensure that you have successfully installed the Oracle Database with the Oracle RAC software on at least one node in your cluster environment.
2. Ensure that the `$ORACLE_HOME` environment variable identifies the successfully installed Oracle home.
3. Run the `addNode.sh` script

   On an existing node from the *`Oracle_home`*`/oui/bin` directory, run the `addNode.sh` script. This script starts OUI in `add node` mode and displays the OUI Welcome page. Click **Next** on the Welcome page and OUI displays the Specify Cluster Nodes for Node Addition page.
4. Verify the entries that OUI displays.

   The Specify Cluster Nodes for Node Addition page has a table showing the existing nodes associated with the Oracle home from which you launched OUI. A node selection table appears on the bottom of this page showing the nodes that are available for addition. Select the nodes that you want to add and click **Next**.
5. The OUI verifies connectivity and performs availability checks on the existing nodes and on the nodes that you want to add. Some of checks performed determine whether:
   - The nodes are up
   - The nodes are accessible by way of the network
6. If any of the checks fail, then fix the problem and proceed or deselect the node that has the error and proceed. You cannot deselect existing nodes; you must correct problems on the existing nodes before proceeding with node addition. If all of the checks succeed, then OUI displays the Node Addition Summary page.

   Note: If any of the existing nodes are down, then perform the `updateNodeList` procedure on each of the nodes to fix the node list after the nodes are up. Run the following command where *`node_list`* is a comma-delimited list of all of the nodes on which Oracle RAC is deployed:

   ```
   oui/bin/runInstaller -updateNodeList
   "CLUSTER_NODES={node_list}" -local
   ```

   ---

7. The Node Addition Summary page has the following information about the products that are installed in the Oracle home that you are going to extend to the target nodes:
   - The source for the add node process, which in this case is the Oracle home
   - The existing nodes and target nodes
   - The target nodes that you selected
   - The required and available space on the target nodes
   - The installed products listing all of the products that are already installed in the existing Oracle home

   Click **Finish** and OUI displays the Cluster Node Addition Progress page.
8. The Cluster Node Addition Progress page shows the status of the cluster node addition process. The table on this page has two columns showing the four phases of the node addition process and each phases' status as follows:

- Copy the Oracle home to the New Nodes—Copies the entire Oracle home from the local node to the target nodes unless the Oracle home is on a cluster file system
- Save Cluster Inventory—Updates the node list associated with the Oracle home and its inventory
- Run `root.sh`—Displays the dialog prompting you to run `root.sh` on the target nodes

The Cluster Node Addition Progress page's Status column displays `Succeeded` if the phase completes, `In Progress` if the phase is in progress, and `Suspended` when the phase is pending execution. After OUI displays the End of Node Addition page, click **Exit** to end the OUI session.

9. Run the `root.sh` script on all of the target nodes.

   Run the `root.sh` script on the target node from the `Oracle_home` when OUI prompts you to do so.

   Oracle Clusterware issues the `oifcfg` command, as shown in the following example:

   ```
   oifcfg setif -global eth0/146.56.76.0:public eth1/192.0.0.0:cluster_
   interconnect
   ```

   This sets both networks to `global`. Therefore, you do not need to run the `oifcfg` command manually after you add a node unless the network interfaces differ.

10. On the target node, run the Net Configuration Assistant (NETCA) to add a listener.

    Add a listener to the target node by running NETCA from the target node and selecting only the target node on the Node Selection page.

You can now add database instances to the target nodes as described in "Add ASM and Oracle RAC Database Instances to Target Nodes" on page 9-9.

Extending the Database Home to Target Nodes Using OUI in Silent Mode

You can optionally run `addNode.sh` in silent mode, replacing steps 1 through 6, as follows where `nodeI`, `nodeI+1`, and so on are the target nodes to which you are adding the Oracle RAC database home.

1. Ensure that you have successfully installed the Oracle Database with the Oracle RAC software on at least one node in your cluster environment.
2. Ensure that the `$ORACLE_HOME` environment variable identifies the successfully installed Oracle home.
3. Go to *`Oracle_home`*`/oui/bin` and run the `addNode.sh` script. In the following example, `nodeI`, `nodeI+1` (and so on) are the nodes that you are adding:

   ```
   addNode.sh -silent "CLUSTER_NEW_NODES={nodeI, nodeI+1, … nodeI+n}"
   ```

   You can also specify the `variable=value` entries in a response file, known as *`filename`*, and you can run the `addNode` script as follows:

   ```
   addNode.sh -silent -responseFile filename
   ```

   Command-line values always override response file values.

Add ASM and Oracle RAC Database Instances to Target Nodes

You can use either Enterprise Manager or DBCA to add Oracle RAC database instances to the target nodes. To add a database instance to a target node with Enterprise Manager, see the *Oracle Database 2 Day + Real Application Clusters Guide* for complete information.

This section describes using DBCA to add Oracle RAC database instances under the following topics:

- Using DBCA in Interactive Mode to Add ASM and Database Instances to Target Nodes
- Using DBCA in Silent Mode to Add ASM and Database Instances to Target Nodes

These tools guide you through the following tasks:

- Creating and starting an ASM instance (if the existing instances were using ASM) on each target node
- Creating a new database instance on each target node
- Creating and configuring high availability components
- Creating the Oracle Net configuration
- Starting the new instance
- Creating and starting services if you entered services information on the Services Configuration page

After adding the instances to the target nodes, you should perform any necessary service configuration procedures, as described in Chapter 4.

Using DBCA in Interactive Mode to Add ASM and Database Instances to Target Nodes

To add a database instance to a target node with DBCA in interactive mode, perform the following steps:

1. Ensure that your existing nodes have the `$ORACLE_HOME` environment variable set correctly.
2. Start the DBCA by entering `dbca` at the system prompt from the `bin` directory in the *`Oracle_home`* directory.

   The DBCA displays the Welcome page for Oracle RAC. Click **Help** on any DBCA page for additional information.
3. Select **Oracle Real Application Clusters database**, click **Next**, and DBCA displays the Operations page.
4. Select **Instance Management**, click **Next**, and DBCA displays the Instance Management page.
5. Select **Add Instance** and click **Next**. The DBCA displays the List of Cluster Databases page that shows the databases and their current status, such as `ACTIVE`, or `INACTIVE`.
6. From the List of Cluster Databases page, select the active Oracle RAC database to which you want to add an instance. Enter user name and password for the database user that has `SYSDBA` privileges. Click **Next** and DBCA displays the List of Cluster Database Instances page showing the names of the existing instances for the Oracle RAC database that you selected.

7. Click **Next** to add a new instance and DBCA displays the Adding an Instance page.
8. On the Adding an Instance page, enter the instance name in the field at the top of this page if the instance name that DBCA provides does not match your existing instance naming scheme. Then select the target node name from the list, click **Next**, and DBCA displays the Services Page.
9. Enter the services information for the target node's instance, click **Next**, and DBCA displays the Instance Storage page.
10. If you are using raw devices or raw partitions, then on the Instance Storage page select the Tablespaces folder and expand it. Select the undo tablespace storage object and a dialog appears on the right-hand side. Change the default datafile name to the raw device name for the tablespace.
11. If you are using raw devices or raw partitions or if you want to change the default redo log group file name, then on the Instance Storage page select and expand the Redo Log Groups folder. For each redo log group number that you select, DBCA displays another dialog box. Enter the raw device name that you created in the section "Configure Shared Storage on Target Nodes" on page 9-5 in the **File Name** field.
12. If you are using a cluster file system, then click **Finish** on the Instance Storage page. If you are using raw devices, then repeat step 11 for all of the other redo log groups, click **Finish**, and DBCA displays a Summary dialog.
13. Review the information on the Summary dialog and click **OK** or click **Cancel** to end the instance addition operation. The DBCA displays a progress dialog showing DBCA performing the instance addition operation. When DBCA completes the instance addition operation, DBCA displays a dialog asking whether you want to perform another operation.
14. After you terminate your DBCA session, run the following command to verify the administrative privileges on the target node and obtain detailed information about these privileges where *nodelist* consists of the target nodes:

    ```
    cluvfy comp admprv -o db_config -d oracle_home -n nodelist [-verbose]
    ```

15. Perform any needed service configuration procedures, as described in Chapter 4, "Introduction to Automatic Workload Management".

Using DBCA in Silent Mode to Add ASM and Database Instances to Target Nodes

You can use the DBCA in silent mode to add instances to nodes on which you have extended an Oracle Clusterware home and an Oracle Database home. Use the following syntax:

```
dbca -silent -addInstance -nodeList node -gdbName gdbname [-instanceName instname]
 -sysDBAUserName sysdba -sysDBAPassword password
```

Table 9–1 Variables in the DBCA Silent Mode Syntax

Variable	Description
node	The node on which you want to add (or delete) the instance.
gdbname	Global database name.
instname	Name of the instance. Provide an instance name only if you want to override the Oracle naming convention for Oracle RAC instance names.

Table 9–1 (Cont.) Variables in the DBCA Silent Mode Syntax

Variable	Description
sysdba	Name of the Oracle user with SYSDBA privileges.
password	Password for the SYSDBA user.

Before you issue the dbca command, ensure that you have set the $ORACLE_HOME environment variable correctly on the existing nodes.

Deleting Cluster Nodes from Oracle Real Application Clusters Environments

This section provides the following topics that explain the steps you perform to delete nodes from clusters in an Oracle RAC environment:

- Step 1: Delete Instances from Oracle Real Application Clusters Databases
- Step 2: Delete Nodes from the Cluster

Step 1: Delete Instances from Oracle Real Application Clusters Databases

The procedures in this section explain how to use DBCA in interactive or silent mode, to delete an instance from an Oracle RAC database. To delete a database instance from a target node with Enterprise Manager, see the *Oracle Database 2 Day + Real Application Clusters Guide*.

This section includes the following topics:

- Using DBCA in Interactive Mode to Delete Instances from Nodes
- Using DBCA in Silent Mode to Delete Instances from Nodes

> **Note:** If the instance uses ASM, and ASM is installed in a separate *Oracle home*, then using these procedures will *not* remove the ASM instance from the specified node.

Using DBCA in Interactive Mode to Delete Instances from Nodes

To delete an instance using DBCA in interactive mode, perform the following steps:

1. Verify there is a current backup of the OCR.

   Issue the ocrconfig -showbackup command to ensure there is a valid backup.

2. Start DBCA.

   Start DBCA on a node *other than* the node that hosts the instance that you want to delete. The database and the instance that you plan to delete should continue to be started and running during this step.

3. On the DBCA Welcome page select **Oracle Real Application Clusters Database**, click **Next**. DBCA displays the Operations page.

4. On the DBCA Operations page, select **Instance Management** and click **Next**. DBCA displays the Instance Management page.

5. On the DBCA Instance Management page, select the instance to be deleted, select **Delete Instance,** and click **Next**.

6. On the List of Cluster Databases page, select the Oracle RAC database from which to delete the instance, as follows:

   a. On the List of Cluster Database Instances page, DBCA displays the instances that are associated with the Oracle RAC database that you selected and the status of each instance. Select the cluster database from which you will delete the instance.

   b. Enter a user name and password for the database user that has `SYSDBA` privileges. Click **Next**.

   c. Click **OK** on the Confirmation dialog to proceed to delete the instance.

      DBCA displays a progress dialog showing that DBCA is deleting the instance. During this operation, DBCA removes the instance and the instance's Oracle Net configuration. When DBCA completes this operation, DBCA displays a dialog asking whether you want to perform another operation.

      Click **No** and exit DBCA or click **Yes** to perform another operation. If you click **Yes**, then DBCA displays the Operations page.

7. If you have services configured, reassign the services.

   Modify the services so that each service can run on one of the remaining instances. Set "not used" for each service running on the instance that is to be deleted. Click **Finish**.

8. Verify that the dropped instance's redo thread has been removed by querying the `V$LOG` view. If the redo thread is not disabled, then disable the thread. For example:

   ```
   SQL> ALTER DATABASE DISABLE THREAD 2;
   ```

9. Verify that the instance has been removed from the OCR by issuing the following commands:

   ```
   srvctl config database -d database_name
   cd CRS_HOME/bin
   ./crs_stat
   ```

10. If this node had an ASM instance and the node will no longer be a part of the cluster, you must remove the ASM instance by issuing the following commands:

    ```
    srvctl stop asm -n node_name
    srvctl remove asm -n node_name
    ```

    Verify that ASM has been removed by issuing the following command:

    ```
    srvctl config asm -n node_name
    ```

11. If you are deleting more than one node, then repeat these steps to delete the instances from all the nodes that you are going to delete.

Using DBCA in Silent Mode to Delete Instances from Nodes

You can use DBCA in silent mode to delete a database instance from a node.

Issue the following command, where the variables are the same as those shown in Table 9–1 for the DBCA command to add an instance. Provide a node name only if you are deleting an instance from a node other than the one on where DBCA is running:

```
dbca -silent -deleteInstance [-nodeList node] -gdbName gdbname -instanceName
 instname -sysDBAUserName sysdba -sysDBAPassword password
```

At this point, you have accomplished the following:

- Deregistered the selected instance from its associated Oracle Net Services listeners
- Deleted the selected database instance from the instance's configured node
- Removed the Oracle Net configuration
- Deleted the Oracle Flexible Architecture directory structure from the instance's configured node.

Step 2: Delete Nodes from the Cluster

Once you have deleted the instance, you can begin the process of deleting the node from the cluster. You accomplish this by running scripts on the node you want to delete to remove the Oracle Clusterware installation and you run scripts on the remaining nodes to update the node list.

The following steps assume that the node to be removed (`node2` in this discussion) is still functioning. Before beginning these procedures, ensure that the `$ORACLE_HOME` environment variable is set correctly on the existing nodes.

Use the following procedures to delete nodes from Oracle clusters on Linux or UNIX systems:

1. Stop the node applications on the node you are deleting.

   As the `root` user, issue the `srvctl stop nodeapps` command to stop and remove the nodeapps on the node you are deleting. For example:

   ```
   # srvctl stop nodeapps -n nodename
   ```

2. Remove the listener from the node.

   If this is the Oracle home from which the node-specific listener named `LISTENER_nodename` runs, then use NETCA to remove this listener. If necessary, re-create this listener in another home. Invoke NETCA and proceed as follows:

   a. Choose **Cluster Configuration**.

   b. Select only the node you are removing and click **Next**.

   c. Choose **Listener Configuration** and click **Next**.

   d. Choose **Delete** and delete any listeners configured on the node you are removing.

   See Also: *Oracle Database Net Services Administrator's Guide* for more information about NETCA

3. Depending on whether you have a shared or nonshared Oracle home, complete one of the following two procedures:

   - For a shared home, run the following command on each of the nodes that are to be deleted:

     ```
     runInstaller -detachHome -local ORACLE_HOME=Oracle_home
     ```

   - For a nonshared home, on each node that you are deleting, perform the following two steps:

     – Run the following command:

     ```
     runInstaller -updateNodeList ORACLE_HOME=Oracle_home
     CLUSTER_NODES="" -local
     ```

The runInstaller command is located in the directory *Oracle_home*/oui/bin. Using this command does not launch an installer GUI.

– Deinstall the Oracle home from the node that you are deleting by running the following command from the *Oracle_home*/oui/bin directory:

```
./runInstaller -deinstall -silent "REMOVE_HOMES={Oracle_home}" -local
```

4. Verify that all database resources are running on each node that will not be deleted.

   a. Issue the crs_stat command from the *CRS_HOME*/bin directory to verify the other nodes. The following example shows the statistics for the database db_name on the node node2:

   ```
   NAME=ora.db_name.db
   TYPE=application
   TARGET=ONLINE
   STATE=ONLINE on node2
   ```

   b. Ensure that the database resource is not running on a node you are deleting. Issue the crs_relocate command from the *CRS_HOME*/bin directory to perform this as shown in the example:

   ```
   crs_relocate ora.db_name.db
   ```

5. Remove the node applications on the node you are deleting.

   As the root user, issue the following command:

   ```
   # srvctl remove nodeapps -n nodename
   ```

6. Update the node list on the remaining nodes in the cluster.

   a. Issue the following command to set the display environment:

   ```
   DISPLAY=ipaddress:0.0; export DISPLAY
   ```

   b. Define the database Oracle homes ($ORACLE_HOME) in the Oracle inventory for the nodes that are still in the cluster. If there is no $ORACLE_HOME, you can skip this step.

   As the ORACLE user, run the installer with the updateNodeList option on any remaining nodes in the cluster, and include a comma-delimited list of nodes that remain in the cluster:

   ```
   $ORACLE_HOME/oui/bin/runInstaller -updateNodeList ORACLE_HOME=Oracle_Home
    CLUSTER_NODES=node1,node3,node4
   ```

7. Stop Oracle Clusterware on the node you are deleting.

   As the root user on the node that you are deleting, run the rootdelete.sh script in the *CRS_home*/install/ directory to disable the Oracle Clusterware applications that are on the node. For example:

   ```
   # cd CRS_home/install
   # ./rootdelete.sh local nosharedvar nosharedhome
   ```

   Only issue this command *once* and use the nosharedhome argument if you are using a local file system. The nosharedvar option assumes the OCR.LOC file is not on a shared file system. The default for this command is sharedhome that prevents you from updating the permissions of local files such that they can be

removed by the oracle user. If the ocr.loc file is on a shared file system, then issue the *CRS_home*/install/rootdelete.sh remote sharedvar command.

If you are deleting more than one node from your cluster, then repeat this step on each node that you are deleting.

8. Delete the node and remove it from the OCR.

   As the root user on any node that you *are not* deleting:

   a. To determine the node number of any node, issue the olsnodes -n command from the *CRS_home*/bin directory. For example:

      ```
      # olsnodes -n
      node1 1
      node2 2
      ```

   b. Run the rootdeletenode.sh script from the *CRS_home*/install/ directory. The rootdeletenode.sh script calls the clscfg -delete script deletes the node from the Oracle cluster and updates the OCR.

      The following example removes only one node, node2:

      ```
      # cd CRS_home/install
      # ./rootdeletenode.sh node2,2
      ```

      To delete only one node, enter the node name and number of the node that you want to delete with the command *CRS_home*/install/rootdelete.sh node1,node1-number. To delete multiple nodes, issue the command *CRS_home*/install/rootdelete.sh node1,*node1-number*,node2,*node2-number*,... *nodeN*,*nodeN-number*, where *node1* through *nodeN* is a list of the nodes that you want to delete, and *node1-number* through *nodeN-number* represents the node number.

      If you do not perform this step, the olsnodes command will continue to display the deleted node as a part of the cluster.

   c. Confirm that the node has been deleted by issuing the olsnodes command:

      ```
      ./olsnodes -n
      node1 1
      ```

9. Define the CRS home in the Oracle inventory for the nodes that are still in the cluster.

   As the ORACLE user, perform the following steps:

   a. Issue the following command to set up the display environment:

      ```
      DISPLAY=ipaddress:0.0; export DISPLAY
      ```

   b. Issue the runInstaller command from the CRS home and specify a comma-delimited list of nodes that remain in the cluster:

      From the CRS home run the installer with the updateNodeList option on any remaining nodes in the cluster, and include a comma-delimited list of nodes that remain in the cluster:

      ```
      CRS_home/oui/bin/runInstaller -updateNodeList ORACLE_HOME=CRS_home
      CLUSTER_NODES=node1, node3, node4 CRS=TRUE
      ```

10. Delete the Oracle home and the CRS home from the deleted node.

Once the node updates are done, you must manually delete the Oracle home and CRS home from the node that you have deleted. Note: if either of these home directories is located on a shared file system, then skip this step.

a. In the Oracle home directory, issue the following command:

```
$ORACLE_HOME: rm -rf *
```

b. In the CRS home directory, issue the following command:

```
$CRS_HOME : rm -rf *
```

11. Ensure that all initialization scripts and soft links are removed from the deleted node.

For example, as the `root` user on a Linux system, issue the following commands:

```
rm -f /etc/init.d/init.cssd
rm -f /etc/init.d/init.crs
rm -f /etc/init.d/init.crsd
rm -f /etc/init.d/init.evmd
rm -f /etc/rc2.d/K96init.crs
rm -f /etc/rc2.d/S96init.crs
rm -f /etc/rc3.d/K96init.crs
rm -f /etc/rc3.d/S96init.crs
rm -f /etc/rc5.d/K96init.crs
rm -f /etc/rc5.d/S96init.crs
rm -Rf /etc/oracle/scls_scr
```

Optionally, you can also remove the `/etc/oracle` directory, the `/etc/oratab` file, and the Oracle inventory from the deleted node.

12. Optionally, remove additional Oracle homes, ASM homes, or Enterprise Manager homes (if used), from the Oracle inventory on all of the remaining nodes.

On all remaining nodes, run the installer to update the node list. The following example assumes that you have deleted `node2` from the cluster:

```
runInstaller -updateNodeList -local \ORACLE_HOME=$ORACLE_HOME
CLUSTER_NODES=node1,node3,node4
```

13. Verify that you have deleted the node from the cluster.

Run the following command to verify that the node is no longer a member of the cluster and to verify that the Oracle Clusterware components have been removed from this node:

```
cluvfy comp crs -n all [-verbose]
```

The response from this command should not contain any information about the node that you deleted; the deleted node should no longer have the Oracle Clusterware components on it.

> **See Also:** *Oracle Clusterware Administration and Deployment Guide* for more information about enabling and using the CVU

10

Design and Deployment Techniques

This chapter briefly describes database design and deployment techniques for Oracle Real Application Clusters (Oracle RAC) environments. It also describes considerations for high availability and provides general guidelines for various Oracle RAC deployments.

The topics in this chapter include:

- Deploying Oracle Real Application Clusters for High Availability
- General Design Considerations for Oracle Real Application Clusters
- General Database Deployment Topics for Oracle Real Application Clusters

Deploying Oracle Real Application Clusters for High Availability

Many customers implement Oracle RAC to provide high availability for their Oracle Database applications. For true high availability, you must make the entire infrastructure of the application highly available. This requires detailed planning to ensure there are no single points of failure throughout the infrastructure. For example, even though Oracle RAC makes your database highly available, if a critical application becomes unavailable, then your business can be negatively affected. For example, if you choose to use the Lightweight Directory Access Protocol (LDAP) for authentication, then you must make the LDAP server highly available. If the database is up but the users cannot connect to the database because the LDAP server is not accessible, then the entire system is down in the eyes of your users.

For mission critical systems, you must be able to perform failover and recovery, and your environment must be resilient to all types of failures. To reach these goals, start by defining service level requirements for your business. The requirements should include definitions of maximum transaction response time and recovery expectations for failures within the datacenter (such as for node failure) or for disaster recovery (if the entire data center fails). Typically, the service level objective is a target response time for work, regardless of failures. Determine the recovery time for each redundant component. Even though you may have hardware components that are running in an active/active mode, do not assume that losing one component cannot result in downtime for other hardware components while the faulty components are being repaired. Also, when components are running in active/passive mode, perform regular tests to validate the failover time. For example, recovery times for storage channels can take minutes. Ensure that the outage times are within your business' service level agreements, and where they are not, work with the hardware vendor to tune the configuration and settings.

When deploying mission critical systems, the testing should include functional testing, destructive testing, and performance testing. Destructive testing includes the injection

of various faults in the system to test the recovery and to make sure if fits in the service level requirements. It also allows the creation of operational procedures for the production system.

To help you design and implement a mission critical or highly available system, Oracle provides a range of solutions that fit every organization regardless of size. Small workgroups and global enterprises alike are able to extend the reach of their critical business applications. With Oracle and the Internet, applications and their data are now reliably accessible everywhere, at any time. The Oracle Maximum Availability Architecture (MAA) is the Oracle best practices blueprint that is based on proven Oracle high availability technologies and recommendations. The goal of the MAA is to remove the complexity in designing an optimal high availability architecture.

See Also: ■

- *Oracle Database High Availability Overview*
- *Oracle Database High Availability Best Practices*
- Oracle Maximum Availability Architecture (MAA) Web site at

  `http://www.oracle.com/technology/deploy/availability/htdocs/maa.htm`

Best Practices for Deploying Oracle RAC in a High Availability Environment

Applications can take advantage of many Oracle Database, Oracle Clusterware, and Oracle RAC features and capabilities to minimize or mask any failure in the Oracle RAC environment. For example:

- Remove TCP/IP timeouts by using the VIP address to connect to the database.
- Create detailed operational procedures and ensure you have the appropriate support contracts in place to match defined service levels for all components in the infrastructure.
- Take advantage of the Oracle RAC Automatic Workload Management features such as connect time failover, Fast Connection Failover, Fast Application Notification, and the Load Balancing Advisory. See Chapter 4, "Introduction to Automatic Workload Management" for more details.
- Place voting disks on separate volume groups to mitigate outages due to slow I/O throughput. For voting devices, (N*2 + 1) mirrors to survive (N/2 + 1) disk failures.
- Place the OCR with I/O service times in the order of 2 ms or less.
- Tune database recovery using the `FAST_START_MTTR_TARGET` initialization parameter.
- Use ASM to manage database storage.
- Ensure that strong change control procedures are in place.
- Check the surrounding infrastructure for high availability and resiliency, such as LDAP, NIS, and DNS. These entities affect the availability of your Oracle RAC database. If possible, perform a local backup procedure routinely.
- Use Enterprise Manager to administer your entire Oracle RAC environment, not just the Oracle RAC database. Use Enterprise Manager to create and modify services, and to start and stop the cluster database instances and the cluster database. See the *Oracle Database 2 Day + Real Application Clusters Guide* for more information using Enterprise Manager in an Oracle RAC environment.

- Use Recovery Manager (RMAN) to back up, restore, and recover data files, control files, server parameter files (SPFILEs) and archived redo log files. You can use RMAN with a media manager to back up files to external storage. You can also configure parallelism when backing up or recovering Oracle RAC databases. In Oracle RAC, RMAN channels can be dynamically allocated across all of the Oracle RAC instances. Channel failover enables failed operations on one node to continue on another node. You can use RMAN in Oracle RAC from Oracle Enterprise Manager Backup Manager or from a command line. See Chapter 5, "Configuring Recovery Manager and Archiving" for more information about using RMAN.
- If you use sequence numbers, then always use `CACHE` with the `NOORDER` option for optimal sequence number generation performance. With the `CACHE` option, however, you may have gaps in the sequence numbers. If your environment cannot tolerate sequence number gaps, then use the `NOCACHE` option or consider pre-generating the sequence numbers. If your application requires sequence number ordering but can tolerate gaps, then use `CACHE` and `ORDER` to cache and order sequence numbers in Oracle RAC. If your application requires ordered sequence numbers without gaps, then use `NOCACHE` and `ORDER`. This combination has the most negative effect on performance compared to other caching and ordering combinations.
- If you use indexes, then consider alternatives, such as reverse key indexes to optimize index performance. Reverse key indexes are especially helpful if you have frequent inserts to one side of an index, such as indexes that are based on insert date.

Consolidating Multiple Applications in a Database or Multiple Databases in a Cluster

Many people want to consolidate multiple applications in a single database or consolidate multiple databases in a single cluster. Oracle Clusterware and Oracle RAC support both types of consolidation.

Creating a cluster with a single pool of storage managed by ASM provides the infrastructure to manage multiple databases whether they are single instance database or Oracle RAC databases.

With Oracle RAC databases, you can adjust the number of instances and which nodes run instances for a given database, based on workload requirements. Features such as cluster-managed services allow you to manage multiple workloads on a single database or across multiple databases. It is important to properly manage the capacity in the cluster when adding work. The processes that manage the cluster—including processes both from Oracle Clusterware and database—must be able to obtain CPU resources in a timely fashion and must be given higher priority in the system.

Oracle recommends that the number of real time LMS*n* processes on a server is less than or equal to the number of processors. (Note that this is the number of recognized CPUs that includes cores. For example, a dual core CPU is considered to be two CPUs). It is important that you load test your system when adding instances on a node to ensure you have enough capacity to support the workload.

If you are consolidating many small databases into a cluster, you may want to reduce the number of Global Cache Service Processes (LMS*n*) created by the Oracle RAC instance. By default, Oracle calculates the number of processes based on the number of CPUs it finds on the server. This calculation may result in more LMS*n* processes than is needed for the Oracle RAC instance. One LMS process may be sufficient for up to 4 CPUs.

To reduce the number of LMS*n* processes, set the `GC_SERVER_PROCESSES` initialization parameter minimally to a value of 1. Add a process for every four CPUs

needed by the application. In general, it is better to have fewer busy LMS*n* processes. Oracle calculates the number of processes when the instance is started, and you must restart the instance if you want to change the value.

Scalability of Oracle Real Application Clusters

Oracle RAC provides concurrent, transactionally consistent access to a single copy of the data from multiple systems. It provides scalability beyond the capacity of a single server. If your application scales transparently on symmetric multiprocessing (SMP) servers, then it is realistic to expect the application to scale well on Oracle RAC, without the need to make changes to the application code.

Traditionally, when a database server runs out of capacity, it is replaced with a new larger server. As servers grow in capacity, they become more expensive. However, for Oracle RAC databases, you have alternatives for increasing the capacity:

- You can migrate applications that traditionally run on large SMP servers to run on clusters of small servers.
- You can maintain the investment in the current hardware and add a new server to the cluster (or create or add a new cluster) to increase the capacity.

Adding servers to a cluster with Oracle Clusterware and Oracle RAC does not require an outage. As soon as the new instance is started, the application can take advantage of the extra capacity.

All servers in the cluster must run the same operating system and same version of Oracle but the servers do not have to be exactly the same capacity. With Oracle RAC, you can build a cluster that fits your needs, whether the cluster is made up of servers where each server is a two CPU commodity server, to clusters where the servers have 32 or 64 CPUs in each server. The Oracle parallel execution feature allows a single SQL statement to be divided up into multiple processes, where each process completes a subset of work. In an Oracle RAC environment, you can define the parallel processes to run only on the instance where the user is connected or to run across multiple instances in the cluster.

See Also:

- Chapter 7, "Using Cloning to Add ASM and Oracle RAC to Nodes in a New Cluster"
- Chapter 8, "Using Cloning to Extend Oracle RAC to Nodes in the Same Cluster"
- Chapter 9, "Adding and Deleting Oracle RAC from Nodes on Linux and UNIX Systems"

General Design Considerations for Oracle Real Application Clusters

This section briefly describes database design and deployment techniques for Oracle RAC environments. It also describes considerations for high availability and provides general guidelines for various Oracle RAC deployments.

Consider performing the following steps during the design and development of applications that you are deploying on an Oracle RAC database:

1. Tune the design and the application
2. Tune the memory and I/O
3. Tune contention

4. Tune the operating system

> **Note:** If an application does not scale on an SMP system, then moving the application to an Oracle RAC database cannot improve performance.

Consider using hash partitioning for insert-intensive online transaction processing (OLTP) applications. Hash partitioning:

- Reduces contention on concurrent inserts into a single database structure
- Affects sequence-based indexes when indexes are locally partitioned with a table and tables are partitioned on sequence-based keys
- Is transparent to the application

If you hash partitioned tables and indexes for OLTP environments, then you can greatly improve performance in your Oracle RAC database. Note that you cannot use index range scans on an index with hash partitioning.

If you are using sequence numbers, then always use the `CACHE` option. If you use sequence numbers with the `CACHE` option, then:

- Your system may lose sequence numbers
- There is no guarantee of the ordering of the sequence numbers

> **Note:** If your environment cannot tolerate sequence number gaps, then consider pre-generating the sequence numbers or use the `ORDER` and `CACHE` options.

General Database Deployment Topics for Oracle Real Application Clusters

This section describes considerations when deploying Oracle RAC databases. Oracle RAC database performance is not compromised if you do not employ these techniques. If you have an effective single-instance design, then your application will run well on an Oracle RAC database. This section contains the following topics:

- Tablespace Use in Oracle Real Application Clusters
- Object Creation and Performance in Oracle Real Application Clusters
- Node Addition and Deletion and the SYSAUX Tablespace in Oracle RAC
- Distributed Transactions and Oracle Real Application Clusters
- Deploying OLTP Applications in Oracle Real Application Clusters
- Flexible Implementation with Cache Fusion
- Deploying Data Warehouse Applications with Oracle Real Application Clusters
- Data Security Considerations in Oracle Real Application Clusters

Tablespace Use in Oracle Real Application Clusters

In addition to using locally managed tablespaces, you can further simplify space administration by using automatic segment space management (ASSM) and automatic undo management.

ASSM distributes instance workloads among each instance's subset of blocks for inserts. This improves Oracle RAC performance because it minimizes block transfers. To deploy automatic undo management in an Oracle RAC environment, each instance must have its own undo tablespace.

Object Creation and Performance in Oracle Real Application Clusters

As a general rule, only use DDL statements for maintenance tasks and avoid executing DDL statements during peak system operation periods. In most systems, the amount of new object creation and other DDL statements should be limited. Just as in single-instance Oracle databases, excessive object creation and deletion can increase performance overhead.

Node Addition and Deletion and the SYSAUX Tablespace in Oracle RAC

If you add nodes to your Oracle RAC database environment, then you may need to increase the size of the `SYSAUX` tablespace. Conversely, if you remove nodes from your **cluster database**, then you may be able to reduce the size of your `SYSAUX` tablespace.

> **See Also:** Your platform-specific Oracle Real Application Clusters installation guide for guidelines about sizing the `SYSAUX` tablespace for multiple instances.

Distributed Transactions and Oracle Real Application Clusters

If you are running XA Transactions in an Oracle RAC environment and the performance is poor, direct all branches of a tightly coupled distributed transaction to the same instance.

To ensure this, create multiple Oracle Distributed Transaction Processing (DTP) services, with one or more on each Oracle RAC instance. Each DTP service is a singleton service that is available on one and only one Oracle RAC instance. All access to the database server for distributed transaction processing must be done by way of the DTP services. Ensure that all of the branches of a single global distributed transaction use the same DTP service. In other words, a network connection descriptor, such as a TNS name, a JDBC URL, and so on, must use a DTP service to support distributed transaction processing.

> **See Also:** "Services and Distributed Transaction Processing in Oracle Real Application Clusters" on page 4-22 for more details about enabling services and distributed transactions and *Oracle Database Advanced Application Developer's Guide* for more information about distributed transactions in Oracle RAC

Deploying OLTP Applications in Oracle Real Application Clusters

Cache Fusion makes Oracle RAC databases the optimal deployment servers for online transaction processing (OLTP) applications. This is because these types of applications require:

- High availability in the event of failures
- Scalability to accommodate increased system demands
- Load balancing according to demand fluctuations

The high availability features of Oracle and Oracle RAC can re-distribute and load balance workloads to surviving instances without interrupting processing. Oracle

RAC also provides excellent scalability so that if you add or replace a node, then Oracle re-masters resources and re-distributes processing loads.

Flexible Implementation with Cache Fusion

To accommodate the frequently changing workloads of online transaction processing systems, Oracle RAC remains flexible and dynamic despite changes in system load and system availability. Oracle RAC addresses a wide range of service levels that, for example, fluctuate due to:

- Varying user demands
- Peak scalability issues like trading storms (bursts of high volumes of transactions)
- Varying availability of system resources

Deploying Data Warehouse Applications with Oracle Real Application Clusters

This section discusses how to deploy data warehouse systems in Oracle RAC environments by briefly describing the data warehouse features available in shared disk architectures. The topics in this section are:

- Speed-Up for Data Warehouse Applications on Oracle Real Application Clusters
- Parallel Execution in Data Warehouse Systems and Oracle Real Application Clusters

Speed-Up for Data Warehouse Applications on Oracle Real Application Clusters

Oracle RAC is ideal for data warehouse applications because it augments the single instance benefits of Oracle. Oracle RAC does this by maximizing the processing available on all of the nodes that belong to an Oracle RAC database to provide speed-up for data warehouse systems.

The query optimizer considers parallel execution when determining the optimal execution plans. The default cost model for the query optimizer is **CPU+I/O** and the cost unit is **time**. In Oracle RAC, the query optimizer dynamically computes intelligent defaults for parallelism based on the number of processors in the nodes of the cluster. An evaluation of the costs of alternative access paths, table scans versus indexed access, for example, takes into account the degree of parallelism (DOP) available for the operation. This results in Oracle selecting the execution plans that are optimized for your Oracle RAC configuration.

Parallel Execution in Data Warehouse Systems and Oracle Real Application Clusters

Oracle's parallel execution feature uses multiple processes to run SQL statements on one or more CPUs. Parallel execution is available on both single-instance Oracle databases and Oracle RAC databases.

Oracle RAC takes full advantage of parallel execution by distributing parallel processing across all available instances. The number of processes that can participate in parallel operations depends on the DOP assigned to each table or index.

> **See Also:** *Oracle Database Performance Tuning Guide* for more information about the query optimizer

Data Security Considerations in Oracle Real Application Clusters

This section describes the following two Oracle RAC security considerations:

- Transparent Data Encryption and Wallets
- Windows Firewall Considerations

Transparent Data Encryption and Wallets

Wallets used by Oracle RAC instances for Transparent Data Encryption may be one of: a local copy of a common wallet, a common wallet stored on shared media that is directly accessed, or a hardware-security module (HSM) based wallet. In every case, you must open the wallet on each Oracle RAC instance before transparent data encryption can be used.

Deployments where no shared storage exists require that each Oracle RAC node maintain a local wallet.

As an alternative to copying the wallet to each node, you can use a hardware-security module (HSM) device to store the master encryption key. HSM devices are recommended over shared storage on a network drive, because an HSM device automatically clones the master key.

> **Note:** For shared access to a key, Oracle recommends using HSM devices. Oracle does not recommend using a shared copy residing on a network file system because when one instance re-keys the master key, other instances are not notified about the change and use the old master key, which can no longer decrypt the column keys. If you need to re-key an Oracle RAC environment, you must copy the wallet to all of the remaining instances and then close and reopen the wallet on each instance.

After you create and provision a wallet on a single node, you must copy the wallet and make it available to all of the other nodes, as follows:

- For systems using Transparent Data Encryption with encrypted wallets, you can use any standard file transport protocol, though Oracle recommends using a secured file transport.
- For systems using Transparent Data Encryption with obfuscated wallets, file transport through a secured channel is recommended.

To specify the directory in which the wallet must reside, set the `WALLET_LOCATION` or `ENCRYPTION_WALLET_LOCATION` parameter in the `sqlnet.ora` file. The local copies of the wallet need not be synchronized for the duration of Transparent Data Encryption usage until the server key is re-keyed though the `ALTER SYSTEM SET KEY` SQL statement. Each time you issue the `ALTER SYSTEM SET KEY` statement on a database instance, you must again copy the wallet residing on that node and make it available to all of the other nodes. Then, you must close and reopen the wallet on each of the nodes. To avoid unnecessary administrative overhead, reserve re-keying for exceptional cases where you believe that the server master key may have been compromised and that not re-keying it could cause a serious security problem.

> **See Also:** *Oracle Database Advanced Security Administrator's Guide* for more information about creating and provisioning a wallet

Windows Firewall Considerations

By default, all installations of Windows Server 2003 Service Pack 1 and higher enable the Windows Firewall to block virtually all TCP network ports to incoming connections. As a result, any Oracle products that listen for incoming connections on a

TCP port will not receive any of those connection requests, and the clients making those connections will report errors.

Depending upon which Oracle products you install and how they are used, you may need to perform additional Windows post-installation configuration tasks so that the Firewall products are functional on Windows Server 2003.

This section contains these topics:

- Oracle Executables Requiring Firewall Exceptions
- Configuring the Windows Firewall
- Troubleshooting Windows Firewall Exceptions

Oracle Executables Requiring Firewall Exceptions Table 10–1 lists the Oracle Database 10*g* Release 1 (10.1) or later executables that listen on TCP ports on Windows. If they are in use and accepting connections from a remote client computer, then Oracle recommends that you add them to the Windows Firewall exceptions list to ensure correct operation. Except as noted, they can be found in *ORACLE_HOME*\bin.

The RMI registry application and daemon executable listed in Table 10–1 are used by Oracle Ultra Search to launch a remote crawler. They must be added to the Windows Firewall exception list if you are using the Ultra Search remote crawler feature, and if the remote crawler is running on a computer with the Windows Firewall enabled.

Note: If multiple Oracle homes are in use, then several firewall exceptions may be needed for the same executable: one for each home from which that executable loads.

Table 10–1 Oracle Executables Requiring Windows Firewall Exceptions

File Name	Executable Name
CRS_HOME\bin\crsd.exe	OracleCRService
CRS_HOME\bin\evmd.exe	OracleEVMService
CRS_HOME\bin\evmlogger.exe	Event manager logging daemon
CRS_HOME\bin\GuiOracleObjManager.exe	Oracle Object Link Manager (GUI version)
CRS_HOME\bin\ocssd.exe	OracleCSService
CRS_HOME\bin\OracleObjManager.exe	Oracle Object Link Manager (CLI version)
CRS_HOME\bin\racgvip.exe	Virtual Internet Protocol Configuration Assistant
CRS_HOME\cfs\Ocfsfindvol.exe	Oracle Cluster Volume Service
dgmgrl.exe	Data Guard Manager
emagent.exe	Oracle Database Control
extproc.exe	External Procedures
hdodbc.exe	Generic Connectivity
oidldapd.exe	Oracle Internet Directory LDAP Server
omtsreco.exe	Oracle Services for Microsoft Transaction Server
oracle.exe	Oracle Database
ORACLE_HOME\apache\apache\apache.exe	Apache Web Server
ORACLE_HOME\bin\emagent.exe	Enterprise Manager Agent

Table 10–1 (Cont.) Oracle Executables Requiring Windows Firewall Exceptions

File Name	Executable Name
ORACLE_HOME\bin\omtsreco.exe	Oracle services for Microsoft Transaction Server
ORACLE_HOME\bin\ORACLE.EXE	Oracle
ORACLE_HOME\bin\racgimon.exe	RACG
ORACLE_HOME\bin\TNSLSNR.exe	Oracle listener
ORACLE_HOME\jdk\bin\java.exe	Java Virtual Machine
ORACLE_HOME\jdk\bin\rmid.exe	RMI daemon executable
ORACLE_HOME\jdk\bin\rmiregistry.exe	RMI registry application
ORACLE_HOME\jdk\jre\bin\rmiregistry.exe	RMI registry application
ORACLE_HOME\opmn\bin\ons.exe	Oracle Notification Service
ORACLE_HOME\opmn\bin\opmn.exe	Oracle Process Manager
pg4arv.exe	Oracle Procedural Gateway for APPC
pg4mqc.exe	Oracle Procedural Gateway for Websphere MQ
pg4mqs.exe	Oracle Procedural Gateway for Websphere MQ
pg4t4ic.exe	Oracle Procedural Gateway for APPC
tg4drsrv.exe	Oracle Transparent Gateway for DRDA
tg4msql.exe	Oracle Transparent Gateway for MS-SQL Server
tg4sybs.exe	Oracle Transparent Gateway for SYBASE
tg4tera.exe	Oracle Transparent Gateway for Teradata
tnslsnr.exe	Oracle TNS listener
WINDOWS_HOME\system32\drivers\Ocfs.sys	System file for Oracle Cluster File System

Configuring the Windows Firewall Post-installation configuration for the Windows Firewall must be undertaken if *all* of the following conditions are met:

- Oracle server-side components are installed.

  These components include the Oracle Database, network listeners, and any Web servers or services.

- The computer services connections from other computers over a network.

  If no other computers connect to the computer with the Oracle software, then no post-installation configuration steps are required and the Oracle software will function as expected.

- The Windows Firewall is enabled.

  If the Windows Firewall is not enabled, then no post-installation configuration steps are required.

You can configure Windows Firewall by opening specific static TCP ports in the firewall or by creating exceptions for specific executables so that they are able to receive connection requests on any ports they choose. To configure the firewall, choose **Control Panel > Windows Firewall > Exceptions** or enter `netsh firewall add...` at the command line.

Alternatively, Windows will inform you if a foreground application is attempting to listen on a port, and it will ask you if you wish to create an exception for that

executable. If you choose to do so, then the effect is the same as creating an exception for the executable either in the Control Panel or from the command line.

Troubleshooting Windows Firewall Exceptions If you cannot establish certain connections even after granting exceptions to the executables listed in Table 10–1, then follow these steps to troubleshoot the installation:

1. Examine Oracle configuration files (such as `*.conf` files), the Oracle key in the Windows registry, and network configuration files in *ORACLE_HOME*`\network\admin`.
2. Pay particular attention to any executable listed in *ORACLE_HOME*`\network\admin\listener.ora` in a `PROGRAM=` clause. Each of these must be granted an exception in the Windows Firewall, because a connection can be made through the TNS listener to that executable.
3. Examine Oracle trace files, log files, and other sources of diagnostic information for details on failed connection attempts. Log and trace files on the database client computer may contain useful error codes or troubleshooting information for failed connection attempts. The Windows Firewall log file on the server may contain useful information as well.
4. If the preceding troubleshooting steps do not resolve a specific configuration issue on Windows XP Service Pack 2, then provide the output from command `netsh firewall show state verbose=enable` to Oracle Support for diagnosis and problem resolution.

See Also:

- `http://www.microsoft.com/downloads/details.aspx?FamilyID=a7628646-131d-4617-bf68-f0532d8db131&displaylang=en` for information on Windows Firewall troubleshooting
- `http://support.microsoft.com/default.aspx?scid=kb;en-us;875357` for more information on Windows Firewall configuration

11

Monitoring Performance

This chapter describes how to monitor and tune Oracle Real Application Clusters (Oracle RAC) performance. This chapter contains the following topics:

- Overview of Monitoring and Tuning Oracle Real Application Clusters Databases
- Verifying the Interconnect Settings for Oracle Real Application Clusters
- Performance Views in Oracle Real Application Clusters
- Creating Oracle Real Application Clusters Data Dictionary Views with CATCLUST.SQL
- Oracle Real Application Clusters Performance Statistics
- Automatic Workload Repository in Oracle Real Application Clusters Environments
- Active Session History Reports for Oracle RAC
- Monitoring Oracle Real Application Clusters Statistics and Wait Events

Overview of Monitoring and Tuning Oracle Real Application Clusters Databases

This section contains the following topics:

- Monitoring Oracle Real Application Clusters and Oracle Clusterware
- Tuning Oracle Real Application Clusters Databases

See Also:

- *Oracle Database 2 Day + Real Application Clusters Guide*, *Oracle Enterprise Manager Concepts*,
- The Enterprise Manager Online Help
- *Oracle Database 2 Day DBA* for more information about basic database tuning
- *Oracle Database 2 Day + Performance Tuning Guide* for more information about general performance tuning
- *Oracle Clusterware Administration and Deployment Guide* for more information about diagnosing problems for Oracle Clusterware components

Monitoring Oracle Real Application Clusters and Oracle Clusterware

Using Oracle Enterprise Manager is the preferred method for monitoring the Oracle RAC and Oracle Clusterware environment. Oracle Enterprise Manager is an Oracle Web-based integrated management solution for monitoring and administering your computing environment. From any location where you can access a web browser, you can manage Oracle RAC databases, application servers, host computers, and Web applications, as well as related hardware and software. For example, you can monitor your Oracle RAC database performance from your office, home, or a remote site, as long as you have access to a Web browser.

Both Oracle Enterprise Manager Database Control and Oracle Enterprise Manager Grid Control are cluster-aware and provide a central console to manage your cluster database. From the Cluster Database Home page, you can do all of the following:

- View the overall system status, such as the number of nodes in the cluster and their current status. This high-level view capability means that you do not have to access each individual database instance for details if you just want to see inclusive, aggregated information.
- View alert messages aggregated across all the instances with lists for the source of each alert message. An **alert message** is an indicator that signifies that a particular metric condition has been encountered. A **metric** is a unit of measurement used to report the system's conditions.
- Review issues that are affecting the entire cluster as well as those that are affecting individual instances.
- Monitor cluster cache coherency statistics to help you identify processing trends and optimize performance for your Oracle RAC environment. Cache coherency statistics measure how well the data in caches on multiple instances is synchronized. If the data caches are completely synchronized with each other, then reading a memory location from the cache on any instance will return the most recent data written to that location from any cache on any instance.

Enterprise Manager accumulates data over specified periods of time, called collection-based data. Enterprise Manager also provides current data, called real-time data.

Oracle Database 2 Day + Real Application Clusters Guide

- Automatic Database Diagnostic Monitor and Oracle RAC Performance
- The Cluster Database Home Page
- The Interconnects Page
- The Cluster Performance Page

The Cluster Database Home Page

When you log in to Oracle Enterprise Manager using a client browser, the Cluster Database Home page appears where you can monitor the status of both Oracle Clusterware and the Oracle RAC environments. Monitoring can include such things as:

- Notification if there are any VIP relocations
- Status of the Oracle Clusterware on each node of the cluster using information obtained through the Cluster Verification Utility (`cluvfy`)
- Notification if node applications (`nodeapps`) start or stop

- Notification of issues in the Oracle Clusterware alert log for the OCR, voting disk issues (if any), and node evictions

The Cluster Database Home page is similar to a single-instance Database Home page. However, on the Cluster Database Home page, Oracle Enterprise Manager displays the system state and availability. This includes a summary about alert messages and job activity, as well as links to all the database and Automatic Storage Management (ASM) instances. For example, you can track problems with services on the cluster including when a service is not running on all of the preferred instances or when a service response time threshold is not being met.

The Interconnects Page

You can use the Oracle Enterprise Manager Interconnects page to monitor the Oracle Clusterware environment. The Interconnects page shows the public and private interfaces on the cluster, the overall throughput on the private interconnect, individual throughput on each of the network interfaces, error rates (if any) and the load contributed by database instances on the interconnect, including:

- Overall throughput across the private interconnect
- Notification if a database instance is using public interface due to misconfiguration
- Throughput and errors (if any) on the interconnect
- Throughput contributed by individual instances on the interconnect

All of this information also is available as collections that have a historic view. This is useful in conjunction with cluster cache coherency, such as when diagnosing problems related to cluster wait events. You can access the Interconnects page by clicking the Interconnect tab on the Cluster Database home page or clicking the Interconnect Alerts link under Diagnostic Findings on the Oracle RAC database home page.

The Cluster Performance Page

Also, the Oracle Enterprise Manager Cluster Database Performance page provides a quick glimpse of the performance statistics for a database. Statistics are rolled up across all the instances in the cluster database in charts. Using the links next to the charts, you can get more specific information and perform any of the following tasks:

- Identify the causes of performance issues.
- Decide whether resources need to be added or redistributed.
- Tune your SQL plan and schema for better optimization.
- Resolve performance issues

The charts on the Cluster Database Performance page include the following:

- **Chart for Cluster Host Load Average**—The Cluster Host Load Average chart in the Cluster Database Performance page shows potential problems that are outside the database. The chart shows maximum, average, and minimum load values for available nodes in the cluster for the previous hour.
- **Chart for Global Cache Block Access Latency**—Each cluster database instance has its own buffer cache in its System Global Area (SGA). Using Cache Fusion, Oracle RAC environments logically combine each instance's buffer cache to enable the database instances to process data as if the data resided on a logically combined, single cache.

- **Chart for Average Active Sessions**—The Average Active Sessions chart in the Cluster Database Performance page shows potential problems inside the database. Categories, called wait classes, show how much of the database is using a resource, such as CPU or disk I/O. Comparing CPU time to wait time helps to determine how much of the response time is consumed with useful work rather than waiting for resources that are potentially held by other processes.
- **Chart for Database Throughput**—The Database Throughput charts summarize any resource contention that appears in the Average Active Sessions chart, and also show how much work the database is performing on behalf of the users or applications. The Per Second view shows the number of transactions compared to the number of logons, and the amount of physical reads compared to the redo size per second. The Per Transaction view shows the amount of physical reads compared to the redo size per transaction. Logons is the number of users that are logged on to the database.

In addition, the **Top Activity** drill down menu on the Cluster Database Performance page enables you to see the activity by wait events, services, and instances. Plus, you can see the details about SQL/sessions by going to a prior point in time by moving the slider on the chart.

For example, the **Cluster Database Performance** page provides a quick glimpse of the performance statistics for an Oracle RAC database. Statistics are rolled up across all of the instances in the cluster database so that users can identify performance issues without going through all the instances. To help triage the performance issues related to services, Enterprise Manager aggregates the activity data at the following levels:

- Aggregate by waits

  All the activity data is presented in 12 categories: CPU, Scheduler, User I/O, System I/O, Concurrency, Application, Commit, Configuration, Administrative, Network, Cluster and Other. The data presented is rolled up from all of the running instances.

- Aggregate by services

  All the activity data is rolled up for each service. When the activity data is presented in this way, it is easy to identify which service is most active, and needs more analysis.

- Aggregate by instances

  As a similar effort, the activity data is rolled up for each instance, if services are not the interested ones.

The aggregates are provided on the pages where the activity data is presented including: Database Performance Page, Top Activity Page, Wait Details Page and Service Details Page.

See Also: *Oracle Database 2 Day + Real Application Clusters Guide*

Tuning Oracle Real Application Clusters Databases

All single-instance tuning practices for the Oracle Database apply to Oracle RAC databases. Therefore, implement the single-instance tuning methodologies described in the *Oracle Database 2 Day + Performance Tuning Guide* and the *Oracle Database Performance Tuning Guide*.

Verifying the Interconnect Settings for Oracle Real Application Clusters

The interconnect and internode communication protocols can affect **Cache Fusion** performance. In addition, the interconnect bandwidth, its latency, and the efficiency of the IPC protocol determine the speed with which Cache Fusion processes block transfers.

To verify the interconnect settings, query the `V$CLUSTER_INTERCONNECTS` and the `V$CONFIGURED_INTERCONNECTS`. For example:

Example 11–1 Verify Interconnect Settings with V$CLUSTER_INTERCONNECTS

```
SQL> SELECT * FROM V$CLUSTER_INTERCONNECTS;

NAME             IP_ADDRESS      IS_ SOURCE
---------------  --------------- --- -------------------------------
eth2             10.137.20.181   NO  Oracle Cluster Repository
```

Example 11–2 Verify Interconnect Settings with V$CONFIGURED_INTERCONNECTS

```
SQL> SELECT * FROM V$CONFIGURED_INTERCONNECTS;

NAME             IP_ADDRESS      IS_   SOURCE
---------------  --------------- ---   -------------------------------
eth2             10.137.20.181   NO    Oracle Cluster Repository
eth0             10.137.8.225    YES   Oracle Cluster Repository

SQL> DESC V$CONFIGURED_INTERCONNECTS
Name                                      Null?    Type
----------------------------------------- -------- ----------------------------
NAME                                               VARCHAR2(15)
IP_ADDRESS                                         VARCHAR2(16)
IS_PUBLIC                                          VARCHAR2(3)
SOURCE                                             VARCHAR2(31)
```

Influencing Interconnect Processing

Once your interconnect is operative, you cannot significantly influence its performance. However, you can influence an interconnect protocol's efficiency by adjusting the IPC buffer sizes.

The Oracle Cluster Registry (OCR) stores your system's interconnect information. Use the `oifcfg getif` command or the `OCRDUMP` utility to identify the interconnect that you are using. You can then change the interconnect that you are using by running an `oifcfg` command.

> **See Also:** Your vendor-specific interconnect documentation for more information about adjusting IPC buffer sizes

Although you *rarely* need to set the `CLUSTER_INTERCONNECTS` parameter, you can use it to assign a private network IP address or NIC as in the following example:

```
CLUSTER_INTERCONNECTS=10.0.0.1
```

If you are using an operating system-specific vendor IPC protocol, then the trace information may not reveal the IP address.

Note: You can also use the `oifcfg` command as described in the *Oracle Clusterware Administration and Deployment Guide* for more information about enabling and using the CVU to assign private network or private IP addresses.

See Also: *Oracle Database Reference* for more information about the `CLUSTER_INTERCONNECTS` parameter

Performance Views in Oracle Real Application Clusters

Each instance has a set of instance-specific views, which are prefixed with `V$`. You can also query global dynamic performance views to retrieve performance information from all of the qualified instances. Global dynamic performance view names are prefixed with `GV$`.

Querying a `GV$` view retrieves the `V$` view information from all qualified instances. In addition to the `V$` information, each `GV$` view contains an extra column named `INST_ID` of data type `NUMBER`. The `INST_ID` column displays the instance number from which the associated `V$` view information was obtained.

You can use the `INST_ID` column as a filter to retrieve `V$` information from a subset of available instances. For example, the following query retrieves the information from the `V$LOCK` view for instances 2 and 5:

```
SQL> SELECT * FROM GV$LOCK WHERE INST_ID = 2 OR INST_ID = 5;
```

See Also: *Oracle Database Reference* for complete descriptions of `GV$` views

Creating Oracle Real Application Clusters Data Dictionary Views with CATCLUST.SQL

If you did not create your Oracle RAC database with the Database Configuration Assistant (DBCA), then you must run the `CATCLUST.SQL` script to create views and tables related to Oracle RAC. You must have `SYSDBA` privileges to run this script.

See Also: The platform-specific Oracle Real Application Clusters installation guides for more information about creating your Oracle RAC database

Oracle Real Application Clusters Performance Statistics

This section provides an overview of the `V$` and `GV$` views that provide statistics that you can use evaluate block transfers in your **cluster**. Use these statistics to analyze interconnect block transfer rates as well as the overall performance of your Oracle RAC database.

Oracle RAC statistics appear as message request counters or as timed statistics. Message request counters include statistics showing the number of certain types of block mode conversions. Timed statistics reveal the total or average time waited for read and write I/O for particular types of operations.

Automatic Workload Repository in Oracle Real Application Clusters Environments

You can use **Automatic Workload Repository (AWR)** to monitor performance statistics related to Oracle RAC databases. AWR automatically generates snapshots of the performance data once every hour and collects the statistics in the workload repository. In Oracle RAC environments, each AWR snapshot captures data from all active instances in the cluster. The data for each snapshot set is captured from the same point in time. AWR stores the snapshot data for all instances in the same table and the data is identified by an instance qualifier. For example, the `BUFFER_BUSY_WAIT` statistic shows the number of buffer waits on each instance. AWR does not store data that is aggregated from across the entire cluster. In other words, the data is stored for each individual instance.

Using the Automatic Database Diagnostic Monitor (ADDM), you can analyze the information collected by AWR for possible performance problems with the Oracle Database. ADDM presents performance data from a cluster-wide perspective, thus enabling you to analyze performance on a global basis. In an Oracle RAC environment, ADDM can analyze performance using data collected from all instances and present it at different levels of granularity, including:

- Analysis for the entire cluster
- Analysis for a specific database instance
- Analysis for a subset of database instances

To perform these analyses, you can run the ADDM Advisor in ADDM for Oracle Real Application Clusters mode to perform an analysis of the entire cluster, in Local ADDM mode to analyze the performance of an individual instance, or in Partial ADDM mode to analyze a subset of instances. You activate ADDM analysis using the advisor framework through Advisor Central in Oracle Enterprise Manager, or through the `DBMS_ADVISOR` and `DBMS_ADDM` PL/SQL packages.

See Also:

- *Oracle Database Performance Tuning Guide* for information about AWR.
- *Oracle Database 2 Day + Real Application Clusters Guide* for more information about how to access and analyze global and local ADDM data using Oracle Enterprise Manager
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the `DBMS_ADVISOR` and `DBMS_ADDM` packages
- *Oracle Database Reference* for more information about the `DBMS_ADDM_FINDINGS` view

Active Session History Reports for Oracle RAC

This section describes Active Session History (ASH) reports for Oracle RAC under the following topics:

- Overview of ASH Reports for Oracle RAC
- ASH Report for Oracle RAC: Top Cluster Events
- ASH Report for Oracle RAC: Top Remote Instance

Overview of ASH Reports for Oracle RAC

ASH is an integral part of the Oracle Database self-management framework and is useful for diagnosing performance problems in Oracle RAC environments. ASH report statistics provide details about Oracle Database session activity. Oracle Database records information about active sessions for all active Oracle RAC instances and stores this data in the System Global Area (SGA). Any session that is connected to the database and using CPU is considered an active session. The exception to this is sessions that are waiting for an event that belongs to the idle wait class.

ASH reports present a manageable set of data by capturing only information about active sessions. The amount of the data is directly related to the work being performed, rather than the number of sessions allowed on the system.

ASH statistics that are gathered over a specified duration can be put into ASH reports. Each ASH report is divided into multiple sections to help you identify short-lived performance problems that do not appear in the ADDM analysis. Two ASH report sections that are specific to Oracle RAC are Top Cluster Events and Top Remote Instance as described in the next two sections.

ASH Report for Oracle RAC: Top Cluster Events

The ASH report Top Cluster Events section is part of the Top Events report that is specific to Oracle RAC. The Top Cluster Events report lists events that account for the highest percentage of session activity in the cluster wait class event along with the instance number of the affected instances. You can use this information to identify which events and instances caused a high percentage of cluster wait events.

ASH Report for Oracle RAC: Top Remote Instance

The ASH report Top Remote Instance section is part of the Top Load Profile report that is specific to Oracle RAC. The Top Remote Instance report shows cluster wait events along with the instance numbers of the instances that accounted for the highest percentages of session activity. You can use this information to identify the instance that caused the extended cluster wait period.

> **See Also:** *Oracle Database Performance Tuning Guide* for more information about ASH reports

Monitoring Oracle Real Application Clusters Statistics and Wait Events

This section explains wait events and statistics specific to Oracle RAC and how to interpret them when assessing performance data generated by the Automatic Workload Repository, Statspack, or by ad-hoc queries of the dynamic performance views.

This section contains the following topics:

- Oracle Real Application Clusters Statistics and Events in AWR and Statspack Reports
- Oracle Real Application Clusters Wait Events
- Monitoring Performance by Analyzing GCS and GES Statistics
- Analyzing Cache Fusion Transfer Impact Using GCS Statistics
- Analyzing Response Times Based on Wait Events

See Also: *Oracle Database Performance Tuning Guide* for more information about wait event analysis and the `spdoc.txt` file for details about the Statspack utility

Oracle Real Application Clusters Statistics and Events in AWR and Statspack Reports

The statistics snapshots generated by AWR and Statspack can be evaluated by producing reports displaying summary data such as load and cluster profiles based on regular statistics and wait events gathered on each instance.

Most of the relevant data is summarized on the Oracle RAC Statistics Page. This information includes:

- Global cache load profile
- Global cache efficiency percentages—workload characteristics
- Global cache and Enqueue Service (GES)—messaging statistics

Additional Oracle RAC sections appear later in the report:

- Global enqueue statistics
- Global CR statistics
- Global `CURRENT` served statistics
- Global cache transfer statistics.

Oracle Real Application Clusters Wait Events

Analyzing and interpreting what sessions are waiting for is an important method to determine where time is spent. In Oracle RAC, the wait time is attributed to an event which reflects the exact outcome of a request. For example, when a session on an instance is looking for a block in the global cache, it does not know whether it will receive the data cached by another instance or whether it will receive a message to read from disk. The wait events for the global cache now convey precise information and waiting for global cache blocks or messages is:

- Summarized in a broader category called Cluster Wait Class
- Temporarily represented by a placeholder event which is active while waiting for a block, for example:
  - gc current block request
  - gc cr block request
- Attributed to precise events when the outcome of the request is known, for example:
  - gc current block 3-way
  - gc current block busy
  - gc cr block grant 2-way

In summary, the wait events for Oracle RAC convey information valuable for performance analysis. They are used in Automatic Database Diagnostic Monitor (ADDM) to enable precise diagnostics of the effect of cache fusion.

Monitoring Performance by Analyzing GCS and GES Statistics

In order to determine the amount of work and cost related to inter-instance messaging and contention, examine block transfer rates, remote requests made by each transaction, the number and time waited for global cache events as described under the following headings:

- Analyzing the Effect of Cache Fusion in Oracle Real Application Clusters
- Analyzing Performance Using GCS and GES Statistics

Analyzing the Effect of Cache Fusion in Oracle Real Application Clusters

The effect of accessing blocks in the global cache and maintaining coherency is represented by

- The Global Cache Service statistics for current and cr blocks, for example, gc current blocks received, gc cr blocks received, and so on
- The Global Cache Service wait events, for gc current block 3-way, gc cr grant 2-way, and so on.

The response time for cache fusion transfers is determined by the messaging and processing times imposed by the physical interconnect components, the IPC protocol and the GCS protocol. It is not affected by disk I/O factors other than occasional log writes. The cache fusion protocol does not require I/O to data files in order to guarantee **cache coherency** and Oracle RAC inherently does not cause any more I/O to disk than a non-clustered instance.

Analyzing Performance Using GCS and GES Statistics

This section describes how to monitor Global Cache Service performance by identifying data blocks and objects which are frequently used ("hot") by all instances. High concurrency on certain blocks may be identified by Global Cache Service wait events and times.

The gc current block busy wait event indicates that the access to cached data blocks was delayed because they were busy either in the remote or the local cache. This means that the blocks were pinned or held up by sessions or delayed by a log write on a remote instance or that a session on the same instance is already accessing a block which is in transition between instances and the current session needs to wait behind it (for example, gc current block busy).

The `V$SESSION_WAIT` view to identify objects and data blocks with contention. The gc wait events contain the file and block number for a block request in p1 and p2, respectively.

An additional segment statistic, gc buffer busy, has been added to quickly determine the "busy" objects without recourse to the query on `V$SESSION_WAIT` mentioned earlier.

The AWR infrastructure provides a view of active session history which can also be used to trace recent wait events and their arguments. It is therefore useful for hot block analysis. Most of the reporting facilities used by AWR and Statspack contain the object statistics and cluster wait class category, so that sampling of the views mentioned earlier is largely unnecessary.

It is advisable to run ADDM on the snapshot data collected by the AWR infrastructure to obtain an overall evaluation of the impact of the global cache. The advisory will also identify the busy objects and SQL highest cluster wait time.

Analyzing Cache Fusion Transfer Impact Using GCS Statistics

This section describes how to monitor Global Cache Service performance by identifying objects read and modified frequently and the service times imposed by the remote access. Waiting for blocks to arrive may constitute a significant portion of the response time, in the same way that reading from disk could increase the block access delays, only that cache fusion transfers in most cases are faster than disk access latencies.

The following wait events indicate that the remotely cached blocks were shipped to the local instance without having been busy, pinned or requiring a log flush:

- gc current block 2-way
- gc current block 3-way
- gc cr block 2-way
- gc cr block 3-way

The object statistics for gc current blocks received and gc cr blocks received enable quick identification of the indexes and tables which are shared by the active instances. As mentioned earlier, creating an ADDM analysis will, in most cases, point you to the SQL statements and database objects that could be impacted by inter-instance contention.

Any increases in the average wait times for the events mentioned earlier could be caused by the following occurrences:

- High load: CPU shortages, long run queues, scheduling delays
- Misconfiguration: using public instead of private interconnect for message and block traffic

If the average wait times are acceptable and no interconnect or load issues can be diagnosed, then the accumulated time waited can usually be attributed to a few SQL statements which need to be tuned to minimize the number of blocks accessed.

The column `CLUSTER_WAIT_TIME` in `V$SQLAREA` represents the wait time incurred by individual SQL statements for global cache events and will identify the SQL which may need to be tuned.

Note: Oracle recommends using ADDM and AWR. However, Statspack is available for backward compatibility. Statspack provides reporting only. You must run Statspack at level 7 to collect statistics related to block contention and segment block waits.

Analyzing Response Times Based on Wait Events

Most global cache wait events that show a high total time as reported in the AWR and Statspack reports or in the dynamic performance views are normal and may present themselves as the top database time consumers without actually indicating a problem. This section describes the most important and frequent wait events that you should be aware of when interpreting performance data.

If user response times increases and a high proportion of time waited is for global cache (gc), then you should determine the cause. Most reports include a breakdown of events sorted by percentage of the total time.

It is useful to start with an ADDM report, which analyzes the routinely collected performance statistics with respect to their impact, and points to the objects and SQL

contributing most to the time waited, and then moves on to the more detailed reports produced by AWR and Statspack.

The most important wait events for Oracle RAC include various categories, including:

- Block-oriented
  - gc current block 2-way
  - gc current block 3-way
  - gc cr block 2-way
  - gc cr block 3-way
- Message-oriented
  - gc current grant 2-way
  - gc cr grant 2-way
- Contention-oriented
  - gc current block busy
  - gc cr block busy
  - gc buffer busy acquire/release
- Load-oriented
  - gc current block congested
  - gc cr block congested

The block-oriented wait event statistics indicate that a block was received as either the result of a 2-way or a 3-way message, that is, the block was sent from either the resource master requiring 1 message and 1 transfer, or was forwarded to a third node from which it was sent, requiring 2 messages and 1 block transfer.

The gc current block busy and gc cr block busy wait events indicate that the local instance that is making the request did not immediately receive a current or consistent read block. The term "busy" in these events' names indicates that the sending of the block was delayed on a remote instance. For example, a block cannot be shipped immediately if Oracle has not yet written the redo for the block's changes to a log file.

In comparison to "block busy" wait events, a gc buffer busy event indicates that Oracle cannot immediately grant access to data that is stored in the local buffer cache. This is because a global operation on the buffer is pending and the operation has not yet completed. In other words, the buffer is busy and all other processes that are attempting to access the local buffer must wait to complete.

The existence of gc buffer busy events also means that there is block contention that is resulting in multiple requests for access to the local block. Oracle must queue these requests. The length of time that Oracle needs to process the queue depends on the remaining service time for the block. The service time is affected by the processing time that any network latency adds, the processing time on the remote and local instances, and the length of the wait queue.

The average wait time and the total wait time should be considered when being alerted to performance issues where these particular waits have a high impact. Usually, either interconnect or load issues or SQL execution against a large shared working set can be found to be the root cause.

The message-oriented wait event statistics indicate that no block was received because it was not cached in any instance. Instead a global grant was given, enabling the requesting instance to read the block from disk or modify it.

If the time consumed by these events is high, then it may be assumed that the frequently used SQL causes a lot of disk I/O (in the event of the cr grant) or that the workload inserts a lot of data and needs to find and format new blocks frequently (in the event of the current grant).

The contention-oriented wait event statistics indicate that a block was received which was pinned by a session on another node, was deferred because a change had not yet been flushed to disk or because of high concurrency, and therefore could not be shipped immediately. A buffer may also be busy locally when a session has already initiated a cache fusion operation and is waiting for its completion when another session on the same node is trying to read or modify the same data. High service times for blocks exchanged in the global cache may exacerbate the contention, which can be caused by frequent concurrent read and write accesses to the same data.

The load-oriented wait events indicate that a delay in processing has occurred in the GCS, which is usually caused by high load, CPU saturation and would have to be solved by additional CPUs, load-balancing, off loading processing to different times or a new cluster node.For the events mentioned, the wait time encompasses the entire round trip from the time a session starts to wait after initiating a block request until the block arrives.

A

Server Control Utility Reference

This appendix describes how to administer Oracle Real Application Clusters (Oracle RAC) databases and instances using the Server Control Utility (SRVCTL). The topics in this chapter include:

- Overview of SRVCTL for Oracle Real Application Clusters and Oracle Clusterware
- SRVCTL Command Syntax and Options
- SRVCTL General Cluster Database Administration Tasks
- SRVCTL Cluster Database Configuration Tasks
- SRVCTL Node-Level Tasks
- SRVCTL Command Reference
- SRVCTL Commands

> **See Also:** Your platform-specific Oracle Real Application Clusters installation guide for information about using Database Configuration Assistant (DBCA)

Overview of SRVCTL for Oracle Real Application Clusters and Oracle Clusterware

The Server Control (SRVCTL) utility is installed on each node by default. You can use SRVCTL to start and stop the database and instances, manage configuration information, and to move or remove instances and services. You can also use SRVCTL to add services. SRVCTL also manages configuration information.

Some SRVCTL operations store configuration information in the Oracle Cluster Registry (OCR). SRVCTL performs other operations, such as starting and stopping instances, by sending requests to the Oracle Clusterware process (CRSD), which then starts or stops the Oracle Clusterware resources.

To use SRVCTL, enter the `srvctl` command and its options in case sensitive syntax as described under the heading "SRVCTL Command Reference" on page A-4.

- Guidelines for Using SRVCTL in Oracle Real Application Clusters
- Obtaining Command-Line Help for SRVCTL

Guidelines for Using SRVCTL in Oracle Real Application Clusters

Guidelines for using SRVCTL are:

- To use SRVCTL to change your Oracle RAC database configuration, log in to the database as the owner of the home that you want to manage.

  For example, if you installed the Oracle Database and ASM in separate homes and each one has a different owner, then log in as the database user to manage databases and log in as the ASM user to manage ASM instances. Users who are members of the dba group can start and stop the database.
- Only use the version of SRVCTL that is provided with Oracle Database 11*g* on Oracle RAC databases that are created or upgraded for Oracle Database 11*g*. The version of SRVCTL must be the same as the version of the object (listeners, ASM instances, Oracle RAC databases and their instances, and services) being managed. To ensure the versions are the same release, issue SRVCTL commands from the Oracle home of the database or object you are managing.
- Always issue SRVCTL commands from the Oracle home of the database (or object) that you are administering. You must issue SRVCTL commands from the bin directory in the Oracle Clusterware home when performing any operation related to node applications (for example, srvctl add nodeapps), and from the respective home of the Oracle Listeners, ASM instances, Oracle RAC databases and their instances, and Services when performing any operation on those objects.
- SRVCTL does not support concurrent executions of commands on the same object. Therefore, only run one SRVCTL command at a time for each database, service, or other object.

Obtaining Command-Line Help for SRVCTL

To see help for all SRVCTL commands, from the command line enter:

```
srvctl -h
```

To see the command syntax and a list of options for each SRVCTL command, from the command line enter:

```
srvctl command (or verb) object (or noun) -h
```

To see the SRVCTL version number enter:

```
srvctl -V
```

> **Caution:** Although you may be able to cancel running SRVCTL commands by entering Control-C at the command line, you may corrupt your configuration data by doing this. You are strongly advised not to attempt to terminate SRVCTL in this manner.

SRVCTL Command Syntax and Options

SRVCTL commands, objects, and options are case sensitive. Database, instance, and service names are case insensitive and case preserving. SRVCTL interprets the following command syntax:

```
srvctl command object [options]
```

In SRVCTL syntax:

- srvctl is the command to start the SRVCTL utility.
- *command* is a verb such as start, stop, or remove.

- *object* is an object or target on which SRVCTL performs the command, such as *database* or *instance*. You can also use object abbreviations.
- *options* extend the use of a preceding command combination to include additional parameters for the command. For example, the -i option indicates that a comma-delimited list of preferred instance names follows; sometimes the -i option only permits one value and not a list of names. The -n option indicates that a node name or a comma-delimited list of node names follows.

Note: Enclose comma-delimited lists in double-quote ("...,...") symbols.

SRVCTL Cluster Database Configuration Tasks

The database configuration tasks are:

- Add, modify, and delete **cluster database** configuration information.
- Add an instance or a service to, and delete an instance or service from the configuration of a **cluster** database.
- Move instances and services in a cluster database configuration and modify service configurations.
- Set and unset the environment for an instance or service in a cluster database configuration.
- Set and unset the environment for an entire cluster database in a cluster database configuration.

SRVCTL General Cluster Database Administration Tasks

The general database administration tasks are:

- Start and stop cluster databases
- Start and stop cluster database instances
- Start, stop, and relocate cluster database services
- Obtain statuses of cluster databases, cluster database instances, or cluster database services

SRVCTL Node-Level Tasks

The node-level tasks are:

- Adding and deleting node level applications.
- Setting and unsetting the environment for node-level applications.
- Administering node applications.
- Administering ASM instances.
- Starting and stopping a group of programs that includes virtual IP addresses, listeners, and Oracle Notification Services.

See Also: *Oracle Clusterware Administration and Deployment Guide* for information about Oracle Clusterware node applications (nodeapps)

SRVCTL Command Reference

This section presents the SRVCTL command reference:

- srvctl add
- srvctl config
- srvctl enable
- srvctl disable
- srvctl start
- srvctl stop
- srvctl modify
- srvctl relocate
- srvctl status
- srvctl getenv
- srvctl setenv and unsetenv
- srvctl remove

SRVCTL Commands

This section summarizes the SRVCTL commands, objects, and options. Oracle recommends that you use DBCA to create your Oracle RAC database. This is because DBCA configures both the Oracle Clusterware resources and the Net Service entries for each service.

SRVCTL Commands Summary

Table A–1 summary the SRVCTL commands. Execute SRVCTL commands from the command line and specify one or more objects with the appropriate options for the command and its objects.

Table A–1 SRVCTL Commands Summary

Command	Description
srvctl add on page A-5	Adds the node applications, database, database instance, ASM instance, or service.
srvctl remove on page A-9	Removes the node applications, database, database instance, ASM instance, or service.
srvctl config on page A-9	Lists the configuration for the node applications, database, ASM instance, or service.
srvctl enable on page A-11	Enables the database, database instance, ASM instance, or service.
srvctl disable on page A-13	Disables the database, database instance, ASM instance, or service.
srvctl start on page A-15	Starts the node applications, database, database instance, ASM instance, or service.
srvctl stop on page A-18	Stops the node applications, database, database instance, ASM instance, or service.
srvctl modify on page A-18	Modifies the node applications, database, database instance, or service configuration.

Table A–1 (Cont.) SRVCTL Commands Summary

Command	Description
srvctl relocate on page A-25	Relocates the service from one instance to another.
srvctl status on page A-26	Obtains the status of the node applications, database, database instance, ASM instance, or service.
srvctl getenv on page A-28	Displays the environment variable in the configuration for the node applications, database, database instance, or service.
srvctl setenv and unsetenv on page A-30	Sets and unsets the environment variable in the configuration for the node applications, database, database instance, or service.

SRVCTL Objects Summary

Table A–2 lists the SRVCTL objects for SRVCTL commands. Use the full name or the abbreviation for the purpose described.

Table A–2 SRVCTL Objects (Nouns) and Abbreviations

Object Noun Name	Abbreviation	Purpose
asm	asm	To add, configure, enable, start, obtain the status of, stop, disable, and remove ASM instances.
database	db	To add, configure, modify, manage environment variables for, enable, start, obtain the status of, stop, disable, and remove databases.
instance	inst	To add, configure, modify, manage environment variables for, enable, start, obtain the status of, stop, and remove database instances.
listener	lsnr	To add, configure, modify, start, stop, and remove listeners.
nodeapps	no abbreviation	To add, configure, modify, manage environment variables for, start, obtain the status of, stop, and remove node applications.
service	serv	To add, configure, modify, manage environment variables for, enable, start, obtain the status of, relocate, disable, stop, and remove services from your cluster database.

srvctl add

The SRVCTL add command adds the configuration and the Oracle Clusterware applications to the OCR for the cluster database, named instances, named services, or for the named nodes. To perform srvctl add operations, you must be logged in as the database administrator and be the Oracle account owner on Linux and UNIX systems, or you must be logged on as a user with Administrator privileges on Windows systems.

When adding an instance, the name that you specify with -i must match the ORACLE_SID parameter. The database name given with -d db_unique_name must match the DB_UNIQUE_NAME initialization parameter setting. If DB_UNIQUE_NAME is unspecified, then match the DB_NAME initialization parameter setting. The default setting for DB_UNIQUE_NAME uses the setting for DB_NAME. Also, the domain name given with -m db_domain must match the DB_DOMAIN setting.

Table A–3 srvctl add Summary

Command	Description
srvctl add database on page A-6	Adds a database and configuration.
srvctl add instance on page A-6	Adds one or more instances and configurations.
srvctl add service on page A-7	Adds services.
srvctl add nodeapps on page A-8	Adds node applications.
srvctl add asm on page A-8	Adds ASM instances.
srvctl add listener on page A-9	Adds a listener to the node.

srvctl add database

Adds a database configuration to your cluster database configuration.

Syntax and Options Use the `srvctl add database` command with the following syntax:

```
srvctl add database -d db_unique_name -o oracle_home [-m domain_name] [-p
spfile] [-A addr_str] [-r {PRIMARY |
PHYSICAL_STANDBY | LOGICAL_STANDBY | SNAPSHOT_STANDBY}] [-s start_options]
[-n db_name] - y [ automatic | manual ]
```

Table A–4 srvctl add database Options

Syntax	Description
`-d` *db_unique_name*	Unique name for the database.
`-o` *oracle_home*	The Oracle home for the database.
`-m` *domain_name*	The Domain for the database.
`-p` *spfile*	The server parameter file for the database.
`-A` *addr_str*	The database cluster alias (*name*|*ip*/`netmask[/if1[|if2|...]]`).
`-r {PRIMARY | PHYSICAL_STANDBY | LOGICAL_STANDBY | SNAPSHOT_STANDBY}`	The role of the database (primary, physical standby, snapshot standby, or logical standby).
`-s` *start_options*	Startup options for the database.
`-n` *db_name*	The name of the database, where it is different from the unique name given by the `-d` option.
`-y`	Management policy for the database, either `automatic` or `manual`.

Example An example of this command is:

```
srvctl add database -d crm -o /ora/ora11
```

srvctl add instance

Adds a configuration for an instance to your cluster database configuration.

Syntax and Options Use the `srvctl add instance` command with the following syntax:

```
srvctl add instance -d db_unique_name -i inst_name -n node_name
```

Table A–5 srvctl add instance Options

Option	Description
-d *db_unique_name*	Unique name for the database.
-i *inst_name*	The instance name.
-n *node_name*	The node name.

Examples Examples of this command are:

```
srvctl add instance -d crm -i  crm01 -n gm01
srvctl add instance -d crm -i  crm02 -n gm02
srvctl add instance -d crm -i  crm03 -n gm03
```

srvctl add service

Adds services to a database and assigns them to instances. If you have multiple instances of a cluster database on the same node, then always use only one instance on that node for all of the services that node manages.

Syntax and Options Use the `srvctl add service` command with the following syntax:

```
srvctl add service -d db_unique_name -s service_name -r preferred_list
[-a available_list] [-P TAF_policy]
```

Table A–6 srvctl add service Options

Option	Description
-d *db_unique_name*	Unique name for the database.
-s *service_name*	The service name.
-r *preferred_list*	The list of preferred instances.
-a *available_list*	The list of available instances
-P *TAF_policy*	The TAF policy (NONE, BASIC, or PRECONNECT). If you choose PRECONNECT, it creates a service_Preconnect.

Use the following syntax to add a new preferred or available instance to the service configuration:

```
srvctl add service -d db_unique_name -s service_name
-u [-r new_preferred_inst | -a new_available_inst]
```

Table A–7 srvctl add service Options for a New Instance

Option	Description
-d *db_unique_name*	Unique name for the database.
-s *service_name*	The service name.
-u	Update a new instance-to-service configuration.
-r *new_preferred_inst*	Name of the new preferred instance.
-a *new_available_inst*	Name of new available instance.

Examples Use this example syntax to add a named service to a database with preferred instances in list one and available instances in list two, using basic failover for the available instances:

```
srvctl add  service -d crm -s sales -r crm01,crm02 -a crm03
```

Use this example syntax to add a named service to a database with preferred instances in list one and available instances in list two, using preconnect failover for the available instances:

```
srvctl add  service -d crm -s sales -r crm01,crm02 -a crm03 -P Preconnect
```

srvctl add nodeapps

Adds a node application configuration to the specified node.

> **Note:** On Linux and UNIX systems, you must be logged in as `root` and on Windows, you must be logged in as a user with Administrator privileges to run this command.

Syntax and Options Use the `srvctl add nodeapps` command with the following syntax:

```
srvctl add nodeapps -n node_name -o oracle_home -A addr_str
```

Table A–8 srvctl add nodeapps Options

Option	Description
`-n` *node_name*	Node name.
`-o` *oracle_home*	Oracle Clusterware home for the cluster database.
`-A` *addr_str*	The node level VIP address (*name*|*ip*/`netmask[/if1[|if2|...]]`).

Example An example of this command is:

```
srvctl add nodeapps -n crmnode1 -o /ora/ora11 -A 1.2.3.4/255.255.255.0
```

srvctl add asm

Adds a record for an ASM instance to the specified node.

Syntax and Options Use the `srvctl add asm` command with the following syntax:

```
srvctl add asm -n node_name -i +asm_instance_name -o oracle_home
```

Table A–9 srvctl add asm Options

Option	Description
`-n` *node_name*	Node name.
`-i` *+asm_instance_name*	The ASM instance name.
`-o` *oracle_home*	Oracle home for the cluster database.

Example An example of this command is:

```
srvctl add asm -n crmnode1 -i +asm1 -o /ora/ora11
```

srvctl add listener

Adds a listener to the node.

Syntax and Options Use the srvctl add listener command with the following syntax:

```
srvctl add listener -n node_name -o Oracle_home [-l listener_name]
```

Table A–10 srvctl add listener Options

Option	Description
`-n node_name`	Node name.
`-o oracle_home`	Oracle home for the cluster database.
`-l listener_name`	Displays the listener configuration.

Example The following command adds a listener to the node `node1`:

```
srvctl add listener -n node1 -i -o /ora/ora11
```

srvctl config

The `srvctl config` command displays the configuration stored in the OCR.

Table A–11 srvctl config Summary

Command	Description
srvctl config database on page A-9	Displays the configuration information of the cluster database.
srvctl config service on page A-10	Displays the configuration information for the services.
srvctl config nodeapps on page A-10	Displays the configuration information for the node applications.
srvctl config asm on page A-11	Displays the configuration for the ASM instances on the node.
srvctl config listener on page A-11	Displays a list of configured listeners that are registered with Oracle Clusterware on a given node.

srvctl config database

Displays the configuration for an Oracle RAC database or lists all configured databases.

Syntax and Options Use the `srvctl config database` command with the following syntax:

```
srvctl config database [-d db_unique_name [-a] [-t]]
```

Table A–12 srvctl config database Options

Option	Description
`-d db_unique_name`	Unique name for the database
`-a`	Additional attributes
`-t`	Display sample TNS entries

Examples An example of this command to list all database is:

```
srvctl config database
```

An example of this command to show sample TNS entries for a specific database is:

```
srvctl config database -d db_erp -t
```

srvctl config service

Displays the configuration for a service.

Syntax and Options Use the `srvctl config service` command with the following syntax:

```
srvctl config service -d db_unique_name [-s service_name] [-a]
```

Table A–13 srvctl config service Options

Option	Description
`-d db_unique_name`	Unique name for the database.
`-s service_name`	Service name.
`-a`	Additional attributes.

Example An example of this command is:

```
srvctl config service -d crm -s crm
```

> **Note:** If you do not specify `-s service`, then `SRVCTL` displays all services for the specified cluster database.

srvctl config nodeapps

Displays the configuration for node applications.

Syntax and Options Use the `srvctl config nodeapps` command with the following syntax:

```
srvctl config nodeapps -n node_name [-a] [-g] [-s] [-l]
```

Table A–14 srvctl config nodeapps Option

Option	Description
`-n node_name`	Node name.
`-a`	Displays the VIP address configuration.
`-g`	Displays the GSD configuration.
`-s`	Displays the ONS configuration.
`-l`	Displays the listener configuration.

Example An example of this command is:

```
srvctl config nodeapps -n mynode1
```

srvctl config asm

Displays the configuration for the ASM instances.

Syntax and Options Use the `srvctl config asm` command with the following syntax:

```
srvctl config asm -n node_name
```

The only option available for this command is `-n` to specify the node name.

Example An example of this command is:

```
srvctl config asm -n mynode1
```

srvctl config listener

Displays a list of configured listeners that are registered with Oracle Clusterware on a given node.

Syntax and Options Use the `srvctl config listener` command with the following syntax:

```
srvctl config listener -n node_name
```

The only option available for this command is `-n` that you use to specify the node name.

Example An example of this command is:

```
srvctl config listener -n mynode1
```

srvctl enable

The `SRVCTL` enable command enables the named object so that it can run under Oracle Clusterware for automatic startup, failover, or restart. The Oracle Clusterware application supporting the object may be up or down to use this function. The default value is `enable`. If the object is already enabled, then the command is ignored. Enabled objects can be started, and disabled objects cannot be started.

Table A–15 srvctl enable Summary

Command	Description
srvctl enable database on page A-11	Enables the database.
srvctl enable instance on page A-12	Enables the instance.
srvctl enable service on page A-12	Enables a service.
srvctl enable asm on page A-13	Enables an ASM instance.

srvctl enable database

Enables the Oracle Clusterware resources for a database and enables the database's instances if the database was previously disabled.

Syntax and Options Use the `srvctl enable database` command with the following syntax:

```
srvctl enable database -d db_unique_name
```

The only option available for this command is `-d` to specify the database name.

Example An example of this command is:

```
srvctl enable database -d crm
```

srvctl enable instance

Enables an instance for Oracle Clusterware. If all instances are disabled, then enabling an instance also enables the database.

Syntax and Options Use the `srvctl enable instance` command with the following syntax:

```
srvctl enable instance -d db_unique_name -i inst_name_list
```

Table A–16 srvctl enable instance Option

Option	Description
`-d db_unique_name`	Unique name for the database
`-i inst_name_list`	Comma-delimited list of instance names.

Example An example of this command is:

```
srvctl enable instance  -d crm -i "crm1,crm2"
```

srvctl enable service

Enables a service for Oracle Clusterware. Enabling an entire service also affects the enabling of the service over all of the instances by enabling the service at each one. When the entire service is already enabled, an srvctl enable service operation does not affect all of the instances and enable them. Instead, this operation returns an error. Therefore, you cannot always use the entire set of service operations to manipulate the service indicators for each instance.

Syntax and Options Use the `srvctl enable service` command with the following syntax:

```
srvctl enable service -d db_unique_name {-s service_name_list | -s service_name -i
inst_name}
```

Table A–17 srvctl enable service Options

Option	Description
`-d db_unique_name`	Unique name for the database.
`-s service_name_list`	Comma-delimited list of service names.
`-s service_name`	Single service name.
`-i inst_name`	Instance name.

Examples The following example globally enables a service:

```
srvctl enable service -d crm -s crm
```

The following example enables a service to use a preferred instance:

```
srvctl enable service -d crm -s crm -i crm1
```

srvctl enable asm

Enables an ASM instance.

Syntax and Options Use the `srvctl enable asm` command with the following syntax:

```
srvctl enable asm -n node_name [-i +asm_inst_name]
```

Table A–18 srvctl enable asm Option

Option	Description
`-n node_name`	Node name
`-i inst_name`	ASM instance name.

Example An example of this command is:

```
srvctl enable asm -n crmnode1 -i +asm1
```

srvctl disable

Disables a specified object (cluster database, database instance, ASM instance, or service). The `srvctl disable` command is intended to be used when an object is to be repaired or is down for maintenance to prevent inappropriate automatic restarts. When you issue the `disable` command, the object is disabled and unavailable to run under Oracle Clusterware for automatic startup, failover, or restart. If you specify `-i instance_name`, then `SRVCTL` only disables the service from running on the specified instance.

Table A–19 srvctl disable Summary

Command	Description
srvctl disable database on page A-13	Disables the cluster database
srvctl disable instance on page A-14	Disables an instance
srvctl disable service on page A-14	Disables a service
srvctl disable asm on page A-14	Disables an ASM instance

srvctl disable database

Disables a cluster database and its instances.

Syntax and Options Use the `srvctl disable database` command with the following syntax:

```
srvctl disable database -d db_unique_name
```

The only option available for this command is `-d` to specify the database name.

Example An example of this command is:

```
srvctl disable database -d mybd1
```

srvctl disable instance

Disables an instance. If the instance that you disable with this command is the last enabled instance, then this operation also disables the database.

Syntax and Options Use the `srvctl disable instance` command with the following syntax:

```
srvctl disable instance -d db_unique_name -i inst_name_list
```

Table A–20 srvctl disable instance Options

Option	Description
`-d db_unique_name`	Unique name for the database.
`-i inst_name_list`	Comma-delimited instance names.

Example An example of this command is:

```
srvctl disable instance -d crm -i "crm1,crm3"
```

srvctl disable service

Disables a service. Disabling an entire service affects all of the instances, disabling each one. When the entire service is already disabled, a `srvctl disable service` operation on the entire service affects all of the instances and disables them; it just returns an error. This means that you cannot always use the entire set of service operations to manipulate the service indicators for each instance.

Syntax and Options Use the `srvctl disable service` command with the following syntax:

```
srvctl disable service -d db_unique_name {-s service_name_list | -s service_name
-i inst_name}
```

Table A–21 srvctl disable service Options

Option	Description
`-d db_unique_name`	Unique name for the database.
`-s service_name_list`	Comma-delimited service names.
`-s service_name`	Single service name.
`-i inst_name`	Instance name.

Examples The following example globally disables two services:

```
srvctl disable service -d crm -s crm,marketing
```

The following example disables a service running on the preferred named instance and results in running a service on one less instance:

```
srvctl disable service -d crm -s crm -i crm1
```

srvctl disable asm

Disables an ASM instance.

Syntax and Options Use the `srvctl disable asm` command with the following syntax:

```
srvctl disable asm -n node_name [-i +asm_inst_name]
```

Table A–22 srvctl disable asm Option

Option	Description
`-n node_name`	Node name
`-i inst_name`	ASM instance name.

Example An example of this command is:

```
srvctl disable asm -n crmnode1 -i +asm1
```

srvctl start

Starts Oracle Clusterware enabled, non-running applications for the database, all or named instances, all or named service names, or node-level applications. For the `start` command, and for other operations that use a connect string, if you do not provide a connect string, then `SRVCTL` uses "/ as sysdba" to perform the operation. To run such operations, the owner of the `oracle` binary executables must be a member of the OSDBA group, and users running the commands must be in the OSDBA group also.

Table A–23 srvctl start Summary

Command	Description
srvctl start database on page A-16	Starts the cluster database and its instances
srvctl start instance on page A-16	Starts the instance
srvctl start service on page A-16	Starts the service
srvctl start nodeapps on page A-17	Starts the node applications
srvctl start asm on page A-17	Starts ASM instances
srvctl start listener on page A-17	Starts the specified listener or listeners.

srvctl start database

Starts a cluster database and its enabled instances.

Syntax and Options Use the `srvctl start database` command with the following syntax:

```
srvctl start database -d db_unique_name [-o start_options]
```

Table A–24 srvctl start database Options

Option	Description
`-d db_unique_name`	Unique name for the database.
`-o start_options`	Options for startup command (for example: `open`, `mount`, or `nomount`)

Example An example of this command is:

```
srvctl start database -d crm -o open
```

srvctl start instance

Starts instances in the cluster database.

Syntax and Options Use the `srvctl start instance` command with the following syntax:

```
srvctl start instance -d db_unique_name -i inst_name_list [-o start_options]
```

Table A–25 srvctl start instance Options

Option	Description
`-d db_unique_name`	Unique name for the database.
`-i inst_name_list`	Comma-delimited instance names.
`-o start_options`	Options for startup command (for example: open, mount, or nomount).

Example An example of this command is:

```
srvctl start instance -d crm -i "crm1,crm4"
```

srvctl start service

Starts a service or multiple services on the specified instance. The `srvctl start service` command will fail if you attempt to start a service on an instance if that service is already running on its maximum number of instances, that is, its number of preferred instances. You may move a service or change the status of a service on an instance with the srvctl modify service and srvctl relocate service commands described on on page A-22 and on page A-25 respectively.

Syntax and Options Use the `srvctl start service` command with the following syntax:

```
srvctl start service -d db_unique_name [-s service_name_list [-i inst_name]]
[-o start_options]
```

Table A–26 srvctl start service Options

Option	Description
`-d db_unique_name`	Unique name for the database
`-s service_name_list`	Comma-delimited service names; the service name list is optional and if not provided, the `SRVCTL` starts all of the database's services
`-i inst_name`	Instance name
`-o start_options`	Options to startup command (for example: open, mount, or nomount)

Examples The following example starts named service names. If the instances that support these services, including available instances that the service uses for failover, are not running but are enabled, then they are started:

```
srvctl start service -d crm -s crm
```

The following example starts a named service on a specified instance:

```
srvctl start service -d crm -s crm -i crm2
```

srvctl start nodeapps

Starts node-level applications on a particular node.

Syntax and Options Use the `srvctl start nodeapps command` with the following syntax:

```
srvctl start nodeapps -n node_name
```

The only option available for this command is `-n` to specify the node name.

Example An example of this command is:

```
srvctl start nodeapps -n mynode1
```

srvctl start asm

Starts an ASM instance.

Syntax and Options Use the `srvctl start asm` command with the following syntax:

```
srvctl start asm -n node_name [-i +asm_inst_name] [-o start_options]
```

Table A–27 srvctl start asm Option

Option	Description
`-n node_name`	Node name
`-i inst_name`	ASM instance name.
`-o start_options`	Options to startup command, for example `open`, `mount`, or `nomount`.
`-h`	Display help.

Examples An example of this command to start a single ASM instance is:

```
srvctl start asm -n crmnode1 -i +asm1
```

An example to start all ASM instances on a node is:

```
srvctl start asm -n crmnode2
```

srvctl start listener

Starts the default listener known as *node_name*, or starts all of the listeners represented in a given list of listener names, that are registered with Oracle Clusterware on the given node.

Syntax and Options Use the `srvctl start listener` command with the following syntax:

```
srvctl start listener -n node_name [-l listener_name_list]
```

Table A–28 srvctl start listener Options

Option	Description
`-n node_name`	Node name.
`-l listener_name_list`	Displays the listener configuration.

Example An example of this command is:

```
srvctl start listener -n mynode1
```

srvctl stop

Stops the Oracle Clusterware applications for the database, all or named instances, all or named service names, or node level applications. Only the Oracle Clusterware applications that are starting or running are stopped. Objects running outside of Oracle Clusterware are not stopped. Stops node-level applications and all dependent Oracle Clusterware applications on the node.

You should disable an object that you intend to remain stopped after you issue a `srvctl stop` command. See the `SRVCTL disable` command starting with srvctl disable database on page A-13.

> **Note:** If the object is stopped and is not disabled, then it can restart as a result of another planned operation. That is, the object will *not* restart as a result of a failure. Oracle recommends that you disable an object that should remain stopped after you issue a `stop` command.

Table A–29 srvctl stop Summary

Command	Description
srvctl stop database on page A-18	Stops the cluster database
srvctl stop instance on page A-19	Stops the instance
srvctl stop service on page A-19	Stops the service
srvctl stop nodeapps on page A-20	Stops the node-level applications
srvctl stop asm on page A-20	Stops ASM instances
srvctl stop listener on page A-20	Stops the specified listener or listeners.

srvctl stop database

Stops a database, its instances, and its services.

Syntax and Options Use the `srvctl stop database` command with the following syntax:

```
srvctl stop database -d db_unique_name [-o stop_options]
```

Table A–30 srvctl stop database Options

Option	Description
`-d db_unique_name`	Unique name for the database
`-o stop_options`	`shutdown` command options (for example: `normal`, `transactional`, `immediate`, or `abort`)

Example An example of this command is:

```
srvctl stop database -d crm
```

srvctl stop instance

Stops instances and stops all enabled and non-running services that have these instances as either preferred or available instances.

Syntax and Options Use the `srvctl stop instance` command with the following syntax:

```
srvctl stop instance -d db_unique_name -i inst_name_list [-o stop_options]
```

Table A–31 srvctl stop instance Options

Option	Description
`-d db_unique_name`	Unique name for the database
`-i inst_name_list`	Comma-delimited instance names
`-o stop_options`	Options for shutdown command (for example: `normal`, `transactional`, `immediate`, or `abort`)

Example An example of this command is:

```
srvctl stop instance -d crm -i crm1
```

srvctl stop service

Stops one or more services globally across the cluster database, or on the specified instance.

Syntax and Options Use the `srvctl stop service` command with the following syntax:

```
srvctl stop service -d db_unique_name [-s service_name_list [-i inst_name]]
[-f]
```

Table A–32 srvctl stop service Options

Option	Description
`-d db_unique_name`	Unique name for the database
`-s service_name_list`	Comma-delimited service names; if you do not provide a service name list, then SRVCTL stops all services on the database
`-i inst_name`	Instance name

Table A–32 (Cont.) srvctl stop service Options

Option	Description
-f *force*	Force SRVCTL to stop the service; this causes SRVCTL to disconnect all of the sessions transactionally, causing the sessions using the service to reconnect to another instance

Examples The following example stops a service globally across a cluster database:

```
srvctl stop service -d crm -s crm
```

The following example stops a service on a specified instance:

```
srvctl stop  service -d crm -s crm -i crm2
```

srvctl stop nodeapps

Stops node-level applications on a particular node.

Syntax and Options Use the `srvctl stop nodeapps` command with the following syntax:

```
srvctl stop nodeapps -n node_name
```

The only option for this command is the `-n` option, which specifies the node name.

Example An example of this command is:

```
srvctl stop nodeapps -n mynode1
```

srvctl stop asm

Stops an ASM instance.

Syntax and Options Use the `srvctl stop asm` command with the following syntax:

```
srvctl stop asm -n node_name [-i inst_name] [-o stop_options]
```

Table A–33 srvctl stop asm Option

Option	Description
-n node_name	Node name.
-i inst_name	ASM instance name.
-o *stop_options*	Options for shutdown command, for example, `normal`, `transactional`, `immediate`, or `abort`.
-h	Display help.

Example An example of this command is:

```
srvctl stop asm -n crmnode1 -i +asm1
```

srvctl stop listener

Stops the default listener known as *node_name*, or the listeners represented in a given list of listener names, that are registered with Oracle Clusterware on the given node.

Syntax and Options Use the `srvctl stop listener` command with the following syntax:

```
srvctl stop listener -n node_name [-l listener_name_list]
```

Example An example of this command is:

```
srvctl stop listener -n mynode1
```

srvctl modify

Enables you to modify the instance configuration without removing and adding Oracle Clusterware resources. Using `modify` preserves the environment in the OCR configuration that would otherwise need to be re-entered. The configuration description is modified in the OCR configuration, and a new Oracle Clusterware profile is generated and registered. The change takes effect when the application is next restarted.

Table A–34 srvctl modify Summary

Command	Description
srvctl modify database on page A-21	Modifies the configuration for a database.
srvctl modify instance on page A-22	Modifies the configuration for an instance.
srvctl modify service on page A-22	Modifies the configuration for a service.
srvctl modify nodeapps on page A-24	Modifies the configuration for a node application.
srvctl modify listener on page A-25	Modifies the listener configuration on a node.

srvctl modify database

Modifies the configuration for a database.

Syntax and Options Use the `srvctl modify database` command with the following syntax:

```
srvctl modify database -d db_unique_name [-n db_name] [-o oracle_home]
[-m domain_name] [-p spfile] [-r {PRIMARY | PHYSICAL_STANDBY | LOGICAL_STANDBY |
SNAPSHOT_STANDBY}] [-s start_options] [-y {AUTOMATIC | MANUAL}]
```

Table A–35 srvctl modify database Options

Option	Description
`-d db_unique_name`	Unique name for the database.
`-n db_name`	Name of the database, where it is different from the unique name given by -d option.
`-o oracle_home`	Oracle home for cluster database.
`-m domain_name`	Domain for cluster database.
`-p spfile`	Server parameter file for cluster database.
`-r role [PRIMARY | PHYSICAL_STANDBY | LOGICAL_STANDBY | SNAPSHOT_STANDBY]`	Role of the database (primary, physical standby, snapshot standby, or logical standby)
`-s start_options`	Startup options for the database.

Table A–35 (Cont.) srvctl modify database Options

Option	Description
`-y`	Management policy for the database, either `automatic` or `manual`.
`-h`	Print usage.

Example The following example changes the role of a database to a logical standby:

```
srvctl modify database -d crm -r logical_standby
```

srvctl modify instance

Modifies the configuration for a database instance from its current node to another node or changes the dependency between and ASM instance and a database instance.

Syntax and Options Use the `srvctl modify instance` command with the following syntax:

```
srvctl modify instance -d db_unique_name -i inst_name {-n node_name | -s asm_
instance_name | -r}
```

Table A–36 srvctl modify instance Options

Option	Description
`-d` *db_unique_name*	Unique name for the database.
-i inst_name	Database instance name.
`-n` *node_name*	Node name.
`-s` *asm_instance_name*	ASM instance dependency to database instance.
`-r`	Remove ASM instance dependency from database instance.

Examples An example of this command to relocate a database instance is:

```
srvctl modify instance -d crm -i crm1 -n my_new_node
```

The following example of this command establishes a dependency between an ASM instance and a database instance:

```
srvctl modify instance -d crm -i crm1 -s asm1
```

srvctl modify service

Moves a service member from one instance to another. Additionally, this command changes which instances are to be the preferred and the available instances for a service. This command supports some online modifications to the service, for example:

- When a service configuration is modified so that a new preferred or available instance is added, the running state of the existing service is not affected. However, the newly added instances will not automatically provide the service, until a srvctl start service command is issued as described on page A-16.
- When there are available instances for the service, and the service configuration is modified so that a preferred or available instance is removed, the running state of the service may change unpredictably:

- The service is stopped and then removed on some instances according to the new service configuration.
- The service may be running on some instances that are being removed from the service configuration.
- These services will be relocated to the next *free* instance in the new service configuration.

As a result of these considerations, when the online service is being modified, users may experience a brief service outage on some instances even if the instances are not being removed. Or users may experience a brief service outage on instances that are being removed from the service.

> **Important: Oracle recommends that you limit configuration changes to the minimum requirement and that you not perform other service operations while the online service modification is in progress.**

Syntax and Options Use the `srvctl modify service` command with the following syntax:

```
srvctl modify service -d db_unique_name -s service_name -i old_inst_name
-t new_inst_name [-f]
```

Table A–37 srvctl modify service Options for an Available Instance

Option	Description
`-d db_unique_name`	Unique name for the database.
`-s service_name`	Service name.
`-i old_inst_name`	Old instance name.
`-t new_inst_name`	New instance name.
`-f`	Disconnect all sessions during stop or relocate service operations.

You can also use the `srvctl modify service` command to change an available instance to a preferred instance as follows:

```
srvctl modify service -d db_unique_name -s service_name -i avail_inst_name -r [-f]
```

Table A–38 srvctl modify service Options for Changing an Available Instance to Preferred

Option	Description
`-d db_unique_name`	Unique name for the database.
`-s service_name`	Service name.
`-i avail_inst_name`	Instance name.
`-r`	Upgrade instance to preferred.
`-f`	Disconnect all sessions during stop or relocate service operations.

Examples An example of moving a service member from one instance to another is:

```
srvctl modify service -d crm -s crm -i crm1 -t crm2
```

An example of changing an available instance to a preferred instance is:

```
srvctl modify service -d crm -s crm -i crm1 -r
```

To change the status of multiple instances, you can use the `srvctl modify service` command to list which instances are to be the preferred instances and which are to be the available instances for a service, as follows:

```
srvctl modify service -d db_unique_name -s service_name -n -i pref_inst_list [-a
avail_inst_list] [-f]
```

Table A–39 srvctl modify service Options for Changing Instances Between Preferred and Available Status

Option	Description
`-d db_unique_name`	Unique name for the database.
`-s service_name`	Service name.
`-n`	Uses only the instances named for this service (unnamed instances already assigned to the service are removed).
`-i preferred_inst_list`	List of preferred instances.
`-a avail_inst_list`	List of available instances.
`-f`	Disconnect all sessions during stop or relocate service operations.

Example The following command exchanges a preferred and available instance:

```
srvctl modify service -d crm -s crm -n -i crm1 -a crm2
```

srvctl modify nodeapps

Applies a new Oracle home or virtual IP address to nodeapps.

Syntax and Options Use the `srvctl modify nodeapps` command with the following syntax:

```
srvctl modify nodeapps -n node_name [-o oracle_home] [-A new_vip_address]
```

Table A–40 srvctl modify nodeapps Option

Option	Description
`-n node_name`	Node name.
`-o clusterware_home`	Oracle Clusterware home for node applications such as VIP, ONS, and GSD.
`-A new_vip_address`	The node level VIP address (*name*|*ip*/netmask[/if1[|if2|...]]).

Example An example of this command is:

```
srvctl modify nodeapps -n mynode1 -A 100.200.300.40/255.255.255.0/eth0
```

srvctl modify listener

Modifies the default listener configuration known as *node_name*, or the listeners represented in a given list of listener names, that are registered with Oracle Clusterware on the given node.

Syntax and Options Use the `srvctl modify listener` command with the following syntax:

```
srvctl modify listener -n node_name [-l listener_name_list] -o Oracle_home
```

Table A–41 srvctl modify listener Options

Option	Description
`-n node_name`	Node name.
`-o oracle_home`	Oracle home for the cluster database.
`-l lsnr, lsnr ...`	Comma-separated list of the listeners on the node (`[-l lsnr1[lsnr2...]]`).

Example An example of this command is:

```
srvctl modify listener -n mynode1
```

srvctl relocate

Relocates the named service names from one named instance to another named instance. The `srvctl relocate` command works on only one source instance and one target instance at a time, relocating a service from a single source instance to a single target instance. The target instance must be on the preferred or available list for the service. The relocated service is temporary until you modify the configuration. The srvctl modify command described on page A-25 permanently changes the service configuration.

Table A–42 srvctl relocate Summary

Command	Description
srvctl relocate service on page A-25	Relocates the named service names from one named instance to another named instance

srvctl relocate service

Temporarily relocates a service member to run on another instance.

Syntax and Options Use the `srvctl relocate service` command with the following syntax:

```
srvctl relocate service -d db_unique_name -s service_name -i old_inst_name -t new_
inst_name [-f]
```

Table A–43 srvctl relocate service Options

Option	Description
`-d db_unique_name`	Unique name for the database.
`-s service_name`	Service name.

Table A–43 (Cont.) srvctl relocate service Options

Option	Description
`-i old_inst_name`	Old instance name.
`-t new_inst_name`	New instance name.
`-f`	Disconnect all sessions during stop or relocate service operations.

Example To temporarily relocate a named service member from `crm1` to `crm3`:

```
srvctl relocate service -d crm -s crm -i crm1 -t crm3
```

srvctl status

Displays the current state of a named database, instances, services, or node applications.

Table A–44 srvctl status Summary

Command	Description
srvctl status database on page A-26	Obtains the status of a database.
srvctl status instance on page A-26	Obtains the status of a instance.
srvctl status service on page A-27	Obtains the status of services.
srvctl status nodeapps on page A-27	Obtains the status of node applications.
srvctl status asm on page A-28	Obtains the status of ASM instances.

srvctl status database

Obtains the status of instances and their services.

Syntax and Options Use the `srvctl status database` command with the following syntax:

```
srvctl status database -d db_unique_name [-f] [-v]
```

Table A–45 srvctl status database Options

Option	Description
`-d db_unique_name`	Unique name for the database
`-f`	Include disabled applications
`-v`	Verbose output

Example An example of this command is:

```
srvctl status database -d crm -v
```

srvctl status instance

Obtains the status of instances.

Syntax and Options Use the `srvctl status instance` command with the following syntax:

```
srvctl status instance -d db_unique_name -i inst_name_list [-f] [-v]
```

Table A–46 srvctl status instance Options

Option	Description
`-d db_unique_name`	Unique name for the database
`-i inst_name_list`	Comma-delimited list of instance names
`-f`	Include disabled applications
`-v`	Verbose output

Example An example of this command is:

```
srvctl status instance -d crm -i "crm1,crm2" -v
```

srvctl status service

Obtains the status of a service.

Syntax and Options Use the `srvctl status service` command with the following syntax:

```
srvctl status service -d db_unique_name -s service_name_list [-f] [-v]
```

Table A–47 srvctl status service Options

Option	Description
`-d db_unique_name`	Unique name for the database
`-s service_name_list`	Comma-delimited list of service names
`-f`	Include disabled applications
`-v`	Verbose output

Example The following example obtains the status of a named service globally across the database:

```
srvctl status service -d crm -s crm -v
```

srvctl status nodeapps

Obtains the status of node applications on a particular node.

Syntax and Options Use the `srvctl status nodeapps` command with the following syntax:

```
srvctl status nodeapps -n node_name
```

The only option available for this command is `-n` to specify the node name.

Example An example of this command to obtain the status of all nodes supporting database applications is:

```
srvctl status nodeapps -n mynode1
```

srvctl status asm

Obtains the status of an ASM instance.

Syntax and Options Use the `srvctl status asm` command with the following syntax:

```
srvctl status asm -n node_name
```

The only option available for this command is `-n` to specify the node name.

Example An example of this command is:

```
srvctl status asm -n crmnode1
```

srvctl getenv

Gets and displays values for the environment from the configuration file. Use `SRVCTL` with the `set`, `get`, and `unset` environment configuration verbs to administer the environment configurations for databases, instances, services, and node applications.

Table A–48 srvctl getenv Summary

Command	Description
srvctl getenv database on page A-28	Gets the cluster database environment values.
srvctl getenv instance on page A-28	Gets the instance environment values.
srvctl getenv service on page A-29	Gets the service environment values.
srvctl getenv nodeapps on page A-29	Gets the node application environment values.

srvctl getenv database

Displays the cluster database environment values.

Syntax and Options Use the `srvctl getenv database` command with the following syntax:

```
srvctl getenv database -d db_unique_name [-t name_list]
```

Table A–49 srvctl getenv database Options

Options	Description
`-d` *db_unique_name*	Unique name for the database
`-t` *name_list*	Names of environment variables

Example The following example gets the environment configuration for a cluster database:

```
srvctl getenv database -d crm
```

srvctl getenv instance

Gets the values for an instance environment configuration.

Syntax and Options Use the `srvctl getenv instance` command with the following syntax:

```
srvctl getenv instance -d db_unique_name -i inst_name [-t name_list]
```

Table A–50 srvctl getenv database Options

Options	Description
-d *db_unique_name*	Unique name for the database
-i *inst_name*	Instance name
-t *name_list*	Names of environment variables

Example The following example sets the environment configuration for an instance:

```
srvctl getenv instance -d -crm -i instance1
```

srvctl getenv service

Gets the values for a service environment configuration.

Syntax and Options Use the `srvctl getenv service` command with the following syntax:

```
srvctl getenv service -d db_unique_name -s service_name [-t name_list]
```

Table A–51 srvctl getenv service Options

Options	Description
-d *db_unique_name*	Unique name for the database
-s *service_name*	Service name
-t *name_list*	Names of environment variables

Example The following example lists all environment variables for a service:

```
srvctl getenv service -d crm -s crm
```

srvctl getenv nodeapps

Gets the environment variables for the node application configurations.

Syntax and Options Use the `srvctl getenv nodeapps` command with the following syntax:

```
srvctl getenv nodeapps -n node_name [-t name_list]
```

Table A–52 srvctl getenv nodeapps Options

Options	Description
-n *node_name*	Node name
-t name_list	Names of environment variables

Example The following example lists all environment variables for the node applications:

```
srvctl getenv nodeapps -n crmnode1
```

srvctl setenv and unsetenv

The `setenv` command sets values for the environment in the configuration file. The `unsetenv` command unsets values for the environment in the configuration file.

Table A–53 srvctl setenv and unsetenv Summary

Command	Description
srvctl setenv database on page A-30	Administers cluster database environment configurations
srvctl setenv instance on page A-30	Administers instance environment configurations
srvctl setenv service on page A-31	Administers service environment configurations
srvctl setenv nodeapps on page A-31	Administers node application environment configurations
srvctl unsetenv database on page A-32	Unsets the cluster database environment configuration
srvctl unsetenv instance on page A-32	Unsets instance environment configurations
srvctl unsetenv service on page A-33	Unsets service environment configurations
srvctl unsetenv nodeapps on page A-33	Unsets node application environment configurations

srvctl setenv database

Administers cluster database environment configurations.

Syntax and Options Use the `srvctl setenv database` command with the following syntax:

```
srvctl setenv database -d db_unique_name {-t name=val[,name=val,...] | -T
name=val}
```

Table A–54 srvctl setenv database Options

Options	Description
`-d db_unique_name`	Unique name for the database.
`-t =,...nameval`	Names and values of environment variables.
`-T name=val`	Enables single environment variable to be set to a value that contains commas or other special characters.

Example The following example sets the language environment configuration for a cluster database:

```
srvctl setenv database -d crm -t LANG=en
```

srvctl setenv instance

Administers instance environment configurations.

Syntax and Options Use the `srvctl setenv instance` with the following syntax:

```
srvctl setenv instance -d db_unique_name [-i inst_name] {-t
name=val[,name=val,...] | -T name=val}
```

Table A–55 srvctl setenv instance Options

Options	Description
`-d db_unique_name`	Unique name for the database.
`-i inst_name`	Instance name.
`-t =,...nameval`	Names and values of environment variables.
`-T name=val`	Enables single environment variable to be set to a value that contains commas or other special characters.

Example The following example sets the environment configuration for an instance:

```
srvctl setenv instance -d -crm -i instance1 -t LANG=EN
```

srvctl setenv service

Administers service environment configurations.

Syntax and Options Use the `srvctl setenv service` command with the following syntax:

```
srvctl setenv service -d db_unique_name [-s service_name]
{-t name=val[,name=val,...] | -T name=val}
```

Table A–56 srvctl setenv service Options

Options	Description
`-d db_unique_name`	Unique name for the database.
`-s service_name`	Service name.
`-t =,...nameval`	Names and values of environment variables.
`-T name=val`	Enables single environment variable to be set to a value that contains commas or other special characters.

Example To set all environment variables for a service:

```
srvctl setenv service -d crm -s crm -t CLASSPATH=/usr/local/jdk/jre/rt.jar
```

srvctl setenv nodeapps

Sets the environment variables for the node application configurations.

Syntax and Options Use the `srvctl setenv nodeapps` command as follows:

```
srvctl setenv nodeapps -n node_name {-t name=val[,name=value,...] | -T name=value}
```

Table A–57 srvctl setenv nodeapps Options

Options	Description
`-n node_name`	Node name.
`-t =,...nameval`	Names and values of environment variables.
`-T name=val`	Enables single environment variable to be set to a value that contains commas or other special characters.

Example To set an environment variable for a node application:

```
srvctl setenv nodeapps -n crmnode1 -t CLASSPATH=/usr/local/jdk/jre/rt.jar
```

srvctl unsetenv database

Unsets the cluster database environment configurations.

Syntax and Options Use the `srvctl unsetenv database` command as follows:

```
srvctl unsetenv database -d db_unique_name -t name_list
```

Table A–58 srvctl unsetenv database Options

Options	Description
`-d db_unique_name`	Unique name for the database.
`-t name_list`	Names of environment variables.

Example The following example unsets the environment configuration for a cluster database environment variable:

```
srvctl unsetenv database -d crm -t CLASSPATH
```

srvctl unsetenv instance

Unsets instance environment configurations.

Syntax and Options Use the `srvctl unsetenv instance` command as follows:

```
srvctl unsetenv instance -d db_unique_name [-i inst_name] -t name_list
```

Table A–59 srvctl unsetenv instance Options

Options	Description
`-d db_unique_name`	Unique name for the database.
`-i instance_name`	Instance name.
`-t name_list`	Names of environment variables.

Example The following example unsets the environment configuration for an instance:

```
srvctl unsetenv instance -d -crm -i instance1 -t CLASSPATH
```

srvctl unsetenv service

Unsets service environment configurations.

Syntax and Options Use the `srvctl unsetenv service` command as follows:

```
srvctl unsetenv service -d db_unique_name [-s service_name] -t name_list
```

Table A–60 srvctl unsetenv service Options

Options	Description
`-d db_unique_name`	Unique name for the database.
`-s service_name`	Service name.
`-t name_list`	Names of environment variables.

Example To unset an environment variables for a service:

```
srvctl unsetenv service -d crm -s crm -t CLASSPATH
```

srvctl unsetenv nodeapps

Unsets the environment configuration for the node application configurations.

Syntax and Options Use the `srvctl unsetenv nodeapps` command as follows:

```
srvctl unsetenv nodeapps -n node_name -t name_list
```

Table A–61 srvctl unsetenv nodeapps Options

Options	Description
`-n node_name`	Node name.
`-t name=val`	Names and values of environment variables.

Example The following example unsets the environment configuration for a node's node applications:

```
srvctl unsetenv nodeapps -n crmnode1 -t name_list
```

srvctl remove

Removes the configuration, the Oracle Clusterware applications for the node (including the virtual IP address, the Oracle Enterprise Manager agent, the GSD, and the listeners), the database, named instances, or the named services from the cluster database. Environment settings for the object are also removed.

If you do not use the force flag (`-f`), then Oracle prompts you to confirm whether to proceed. If you use the force (`-f`) option, then the remove operation proceeds without prompting and continues processing even when it encounters errors. Even when the Oracle Clusterware resources cannot be removed, the OCR configuration is removed, so that the object now appears not to exist, but there are still Oracle Clusterware resources. Use the force flag (`-f`) option with extreme caution because this could result in an inconsistent OCR.

To use the `remove` verb, you must first stop the node applications, database, instance, or service for which you are specifying `srvctl remove`. Oracle recommends that you perform a disable operation before using this command, but this is not required. You must stop the target object before running the `srvctl remove` command. The srvctl stop command is described on page A-18.

Table A–62 srvctl remove Summary

Command	Description
srvctl remove database on page A-34	Removes a database and configuration.
srvctl remove instance on page A-34	Removes one or more instances and configurations.
srvctl remove service on page A-35	Removes services.
srvctl remove nodeapps on page A-35	Removes node applications.
srvctl remove asm on page A-35	Removes ASM instances
srvctl remove listener on page A-36	Removes the listener from the specified node.

srvctl remove database

Removes a database configuration.

Syntax and Options Use the `srvctl remove database` command with the following syntax:

```
srvctl remove database -d db_unique_name [-f]
```

Table A–63 srvctl remove database Options

Options	Description
`-d db_unique_name`	Unique name for the database.
`-f`	Force remove.

Example An example of this command is:

```
srvctl remove database -d crm
```

srvctl remove instance

Removes the configurations for an instance.

Syntax and Options Use the `srvctl remove instance` command with the following syntax:

```
srvctl remove instance -d db_unique_name -i inst_name [-f]
```

Table A–64 srvctl remove instance Options

Options	Description
`-d db_unique_name`	Unique name for the database.
`-i inst_name`	Instance name.
`-f`	Force remove.

Example An example of this command is:

```
srvctl remove instance -d crm -i crm01
```

srvctl remove service

Removes the configuration for a service.

Syntax and Options Use the `srvctl remove service` command as follows:

```
srvctl remove service -d db_unique_name -s service_name [-i inst_name] [-f]
```

Table A–65 srvctl remove service Options

Options	Description
`-d db_unique_name`	Unique name for the database.
`-s service_name`	Service name.
`-i inst_name`	Instance name.
`-f`	Force remove.

Examples An example of this command is:

```
srvctl remove service -d crm -s sales
```

The following example removes the services from specific instances:

```
srvctl remove service -d crm -s sales -i crm01,crm02
```

srvctl remove nodeapps

Removes the node application configuration from the specified node. You must have full administrative privileges to run this command. On Linux and UNIX systems, you must be logged in as `root` and on Windows systems, you must be logged in as a user with Administrator privileges.

Syntax Use the `srvctl remove nodeapps` command as follows:

```
srvctl remove nodeapps -n node_name_list [-f]
```

Table A–66 srvctl remove nodeapps Options

Options	Description
`-n node_name_list`	Node name or a comma-delimited list of node names.
`-f`	Force remove.

Example An example of this command is:

```
srvctl remove nodeapps -n "mynode1,mynode2,mynode3"
```

srvctl remove asm

Removes an ASM instance.

Syntax and Options Use the `srvctl remove asm` command with the following syntax:

```
srvctl remove asm -n node_name [-i +asm_inst_name]
```

Table A–67 srvctl remove asm Option

Option	Description
`-n node_name`	Node name
`-i +asm_inst_name`	ASM instance name.

Example An example of this command is:

```
srvctl remove asm -n crmnode1 -i +asm1
```

srvctl remove listener

Removes the listener from the specified node.

Syntax and Options Use the `srvctl remove listener` command with the following syntax:

```
srvctl remove listener -n node_name [-l listener_name]
```

Table A–68 srvctl remove listener Options

Options	Description
`-n node_name`	Name of the node.
`-l listener_name`	Name of the listener that you want to remove.

Examples The following command removes the listener `lsnr01` from the `node1` node:

```
srvctl remove listener -n node1 -l lsnr01
```

B

Troubleshooting Oracle Real Application Clusters

This appendix explains how diagnose problems for Oracle Real Application Clusters (Oracle RAC) components using trace and log files. This section includes the following topics:

- Where to Find Files for Analyzing Errors
- Managing Diagnostic Data in Oracle Real Application Clusters
- Using Instance-Specific Alert Files in Oracle Real Application Clusters
- Enabling Tracing for Java-Based Tools and Utilities in Oracle Real Application Clusters
- Resolving Pending Shutdown Issues
- How to Determine If Oracle RAC Instances Are Using the Private Network

> **Note:** The trace and log files generated for the Oracle Database with Oracle RAC are also available for the Oracle Clusterware components. For Oracle Clusterware, Oracle stores these under a unified directory log structure.
>
> See the *Oracle Clusterware Administration and Deployment Guide* for more information about troubleshooting Oracle Clusterware.

Where to Find Files for Analyzing Errors

Oracle records information about important events that occur in your Oracle RAC environment in trace files. The trace files for Oracle RAC are the same as those in single-instance Oracle databases. As a best practice, monitor and back up trace files regularly for all instances to preserve their content for future troubleshooting.

Information about `ORA-600` errors appear in the *`alert_SID`*`.log` file for each instance where *SID* is the instance identifier.

The alert log and all trace files for background and server processes are written to the Automatic Diagnostic Repository, the location of which you can specify with the `DIAGNOSTIC_DEST` initialization parameter. The names of trace files are operating system specific, but each file usually includes the name of the process writing the file (such as LGWR and RECO). For example:

```
diagnostic_dest=/oracle/11.1/diag/rdbms/rac/RAC2/trace
```

Oracle creates a different trace file for each background thread. Oracle RAC background threads use trace files to record database operations and database errors. These trace logs help troubleshoot and also enable Oracle support to more efficiently debug **cluster database** configuration problems. For Linux, UNIX, and Windows systems, trace files for the background processes are named `SID_process name_process identifier.trc`.

> **See Also:** *Oracle Database Administrator's Guide* for more information about monitoring errors and alerts in trace files

Trace files are created for user processes if you set the `DIAGNOSTIC_DEST` initialization parameter. User process trace file names have the format *`SID`*`_ora_`*`process_identifier/thread_identifier`*`.trc`, where *`process_identifier`* is a 5-digit number indicating the process identifier (PID) on Linux and UNIX systems, and *`thread_identifier`* is the thread identifier on Windows systems.

Managing Diagnostic Data in Oracle Real Application Clusters

Problems that span Oracle RAC instances can be the most difficult types of problems to diagnose. For example, you may need to correlate the trace files from across multiple instances, and merge the trace files. Oracle Database Release 11*g* includes an advanced fault diagnosability infrastructure for collecting and managing diagnostic data, and uses the Automatic Diagnostic Repository (ADR) file-based repository for storing the database diagnostic data. When you create the ADR base on a shared disk, you can place ADR homes for all instances of the same Oracle RAC database under the same ADR Base. With shared storage:

- You can use the ADRCI command-line tool to correlate diagnostics across all instances.

  ADRCI is a command-line tool that enables you to view diagnostic data in the ADR and package incident and problem information into a zip file for transmission to Oracle Support. The diagnostic data includes incident and problem descriptions, trace files, dumps, health monitor reports, alert log entries, and so on.

  > **See Also:** *Oracle Database Utilities* for information about using ADRCI

- You can use the Data Recovery Advisor to help diagnose and repair corrupted data blocks, corrupted or missing files, and other data failures.

  The Data Recovery Advisor is an Oracle Database infrastructure that automatically diagnoses persistent data failures, presents repair options, and repairs problems at your request.

  > **See Also:** *Oracle Database Administrator's Guide* for information about managing diagnostic data

Using Instance-Specific Alert Files in Oracle Real Application Clusters

Each instance in an Oracle RAC database has one alert file. The alert file for each instance, alert.SID.log, contains important information about error messages and exceptions that occur during database operations. Information is appended to the alert file each time you start the instance. All process threads can write to the alert file for the instance.

The *alert_SID*.log file is in the directory specified by the DIAGNOSTIC_DEST parameter in the init*db_name*.ora initialization parameter file.

Enabling Tracing for Java-Based Tools and Utilities in Oracle Real Application Clusters

All Java-based tools and utilities that are available in Oracle RAC are invoked by executing scripts of the same name as the tool or utility. This includes the Cluster Verification Utility (CVU), Database Configuration Assistant (DBCA), the Net Configuration Assistant (NETCA), Server Control (SRVCTL), and the Global Services Daemon (GSD). For example to run DBCA, enter the command dbca.

By default, Oracle enables traces for DBCA and the Database Upgrade Assistant (DBUA). For the CVU, GSDCTL, and SRVCTL, you can set the SRVM_TRACE environment variable to TRUE to make Oracle generate traces. Oracle writes traces to log files. For example, Oracle writes traces to log files in *Oracle home*/cfgtoollogs/dbca and *Oracle home*/cfgtoollogs/dbua for DBCA and the DBUA, respectively.

Resolving Pending Shutdown Issues

In some situations a SHUTDOWN IMMEDIATE may be pending and Oracle will not quickly respond to repeated shutdown requests. This is because Oracle Clusterware may be processing a current shutdown request. In such cases, issue a SHUTDOWN ABORT using SQL*Plus for subsequent shutdown requests.

How to Determine If Oracle RAC Instances Are Using the Private Network

This section describes how to manually determine if Oracle RAC instances are using the private network. However, the best practice for this task is to use Oracle Enterprise Manager Database Control graphical user interfaces (GUI) to check the interconnect. Also, see the *Oracle Database 2 Day + Real Application Clusters Guide* for more information about monitoring Oracle RAC using Enterprise Manager.

With most network protocols, you can issue the oradebug ipc command to see the interconnects that the database is using. For example:

```
oradebug setmypid
oradebug ipc
```

These commands dump a trace file to the location specified by the DIAGNOSTIC_DEST initialization parameter. The output may look similar to the following:

```
SSKGXPT 0x1a2932c flags SSKGXPT_READPENDING     info for network 0
        socket no 10    IP 172.16.193.1         UDP 43749
        sflags SSKGXPT_WRITESSKGXPT_UP  info for network 1
        socket no 0     IP 0.0.0.0      UDP 0...
```

In the example, you can see the database is using IP 172.16.193.1 with a User Datagram Protocol (UDP) protocol. Also, you can issue the oradebug tracefile_name command to print the trace location where the output is written.

Additionally, you can query the V$CLUSTER_INTERCONNECTS view to see information about the private interconnect. For example:

```
SQL> SELECT * FROM V$CLUSTER_INTERCONNECTS;

NAME   IP_ADDRESS                IS_   SOURCE
```

```
-----  -------------------------- ---   ------------------------------
eth0   138.2.236.114              NO   Oracle Cluster Repository
```

C

Oracle Real Application Clusters Tools Messages

This appendix describes the Oracle Real Application Clusters (Oracle RAC) management tools messages. The messages in this appendix appear alphabetically by group. The topics in this appendix are:

- Overview of Messages Specific to Oracle Real Application Clusters
- PRKA—Cluster Node Applications Messages
- PRKC—Cluster Command Messages
- PRKD—Global Services Daemon Messages
- PRKE—Global Services Daemon Controller Utility Messages
- PRKH—Server Manager (SRVM) Messages
- PRKI—Cluster Pre-Install Messages
- PRKN—Server Manager (SRVM) System Library Messages
- PRKO—Server Control (SRVCTL) Utility Messages
- PRKP—Cluster Database Management Messages
- PRKR—Cluster Registry Messages
- PRKS—Automatic Storage Management Messages
- PRKU—Command-Line Parser Utility Messages
- PRKV—Virtual IP Configuration Assistant Messages

> **See Also:** *Oracle Database Platform Guide for Microsoft Windows* for Windows messages and for all other messages refer search online at
>
> `http://tahiti.oracle.com`

Overview of Messages Specific to Oracle Real Application Clusters

Oracle RAC messages use the same ORA prefix and reside in the same `e*.msg` files as single-instance Oracle database messages. Additionally, some Oracle RAC messages are issued by the single-instance Oracle database. These messages appear online by way of a Tahiti error message search as described in the Oracle database documentation in the Preface of this book under the heading "Related Documents" on page -x.

Prefixes and Message Codes for Oracle Real Application Clusters Messages

Message prefixes indicate where to find information about the messages in this chapter. In addition, the prefixes indicate which Oracle RAC component issued the messages.

Types of Oracle Real Application Clusters Messages and Related Files

The Oracle RAC high availability and **cluster database** management tools use the following syntax for internal messages:

```
PRKx-0000: Cause description "variable" text {variable} text {variable}
```

A message might appear as follows:

```
PRKC-1022 "Could not get "node name" for node {0} in {1}"
```

The variables in the previous example represent numbers, names, and character strings for objects in the physical or logical cluster and the cluster database. you can ignore empty variables.

PRKA—Cluster Node Applications Messages

PRKA-2001: GSD already exists

Cause: An attempt was made to create the Global Services Daemon (GSD) application while there is a running GSD application.

Action: Stop and remove the running GSD and create the GSD application again using the 'srvctl add nodeapps' command.

PRKA-2002: Listener already exists

Cause: An attempt was made to create a listener application while there is a running listener application.

Action: Stop and remove the running listener and create the listener application again using the 'srvctl add nodeapps' command.

PRKA-2003: EM Agent already exists

Cause: An attempt was made to create an Agent application while there is a running Agent application.

Action: Stop and remove the running Agent and create the Agent application again using the 'srvctl add nodeapps' command.

PRKA-2004: No database found for the cluster.

Cause:

Action:

PRKA-2005: Netmasks are different for starting and ending IP addresses

Cause: Invalid range of VIP addresses specified for the cluster.

Action: Check if the starting and ending addresses in the VIP range belong to the same netmask.

PRKA-2006: Invalid VIP address range

Cause: Invalid range of VIP addresses specified for the cluster.

Action: Check if the starting address is less than ending address.

PRKA-2007: Empty VIP address range

Cause: Invalid range of VIP addresses specified for the cluster.

Action: Enter valid non-null VIP address range.

PRKA-2008: No VIP address is available in the VIP pool

Cause: No IP addresses are available for use in the range of IP addresses that you specified earlier.

Action: Check the VIP address pool to ensure that you are using valid VIP addresses.

PRKA-2009: VIP address is not found

Cause: IP address is not found.

Action: Enter valid VIP address.

PRKA-2010: VIP address already exists

Cause: An attempt was made to create a Virtual IP while there is a running Virtual IP application.

Action: Stop and remove the running Virtual IP application and create the Virtual IP application using the 'srvctl add nodeapps' command.

PRKA-2011: Remote command failed - not used?

Cause: The command failed to execute on remote node.

Action: Check the command for syntax and semantics and reissue the command.

PRKA-2012: Failed to get the VIP address range configuration from the OCR

Cause: There might not be any IP address ranges in the OCR or it is possible that an OCR error occurred while retrieving the VIP address range.

Action: Refer to the OCR logs for detailed error information.

PRKA-2013: Failed to add the VIP address range configuration to the OCR

Cause: An OCR error occurred while adding the VIP address range into the OCR.

Action: Refer to the OCR logs for detailed error information.

PRKA-2014: Failed to remove the VIP address range configuration from the OCR

Cause: An OCR error occurred while adding the VIP address range into the OCR.

Action: Refer to the OCR logs for detailed error information.

PRKA-2015: Invalid node name

Cause: The node name that you entered does not belong to the cluster.

Action: Enter a node that belongs to the cluster.

PRKA-2016: Invalid listener name

Cause: The listener name entered is invalid.

Action: Correct the listener name.

PRKA-2017: There are no IP addresses in the specified range. Make sure to specify the correct range for the IP addresses.

Cause:

Action: Specify valid range for IP addresses.

PRKA-2018: Virtual IP can only be created or removed by the system privilege user.

Cause: Only users with administrative privileges can configure Virtual IPs on the system.

Action: Verify the privileges of the user who is attempting to create or remove the VIP address.

PRKA-2019: Error executing command \"{0}\". File is missing.

Cause:

Action:

PRKC—Cluster Command Messages

PRKC-1023: Invalid IP address format: {0}

Cause: The given IP address does not adhere to the IP address format standard.

Action: Use the correct IP address format.

PRKC-1024: Invalid netmask: {0}

Cause: The given netmask format does not adhere to the format standard.

Action: Change the netmask format to a valid netmask format.

PRKC-1025: Failed to create a file in directory {0}

Cause: Might be due to insufficient permission or due to insufficient disk space.

Action: Check the permission on the directory and available space.

PRKC-1026: Failed doing I/O to file {0}

Cause: Might be due to insufficient permission or due to insufficient disk space or network issue.

Action: Check the permission on the directory and available space. Make sure the storage medium is accessible.

PRKC-1027: Error checking existence of file {0} on {1}

Cause: Might be due to an improper user equivalence configuration or due to insufficient permissions on the parent directory.

Action: Verify that user equivalence is correctly configured and that the directory has read and execute permissions.

PRKC-1028: Error checking write permission for directory {0} on {1}

Cause: Might be due to an improper user equivalence configuration or due to insufficient permissions on the parent directory.

Action: Verify that user equivalence is correctly configured and that the directory has read and execute permissions.

PRKC-1029: Failed to get modification time for file {0} on node {1}, [2]

Cause: Might be due to an improper user equivalence configuration or due to insufficient permissions on the parent directory.

Action: Verify that user equivalence is correctly configured and that the directory has read and execute permissions.

PRKC-1030: Error checking accessibility for node {0}, {1}

Cause: Might be due to a network issue or to an improper user equivalence setup.

Action: Check node reachability between the local and the remote nodes and make sure user equivalence is correctly configured.

PRKC-1031: Error checking free space for {0} on {1}

Cause: Might be due to a network issue or to an improper user equivalence setup.

Action: Check node reachability between the local and the remote nodes and make sure that user equivalence is correctly configured.

PRKC-1032: Directory {0} does not exist

Cause: The specified directory does not exist.

Action: Make sure that the specified directory exists. Use an absolute path for the directory.

PRKC-1033: Executable {0} does not exist

Cause: The specified executable does not exist.

Action: Verify the existence of the executable before retrying.

PRKC-1034: No local node name found for host {0}

Cause: The local node may not be part of the cluster or an error may have occurred while retrieving the local node name.

Action: Check CRS logs in ORA_CRS_HOME/log/<hostname> and OCR logs in ORA_CRS_HOME/srvm/log.

PRKC-1035: Node names for this cluster could not be retrieved

Cause: An error occurred while retrieving node names from the cluster.

Action: Check CRS logs in ORA_CRS_HOME/log/<hostname> and OCR logs in ORA_CRS_HOME/srvm/log.

PRKC-1036: CRS_HOME name passed to the method was null

Cause: The CRS home name cannot be null.

Action: Pass non-null CRS_HOME to the method.

PRKC-1037: Error removing files listed in {0} from node {1}

Cause: Might be due to an improper user equivalence configuration or due to insufficient permissions.

Action: Verify that user equivalence is correctly configured and that the mentioned files have the correct permissions.

PRKC-1038: Error copying files listed in {0} to node {1}

Cause: Might be due to an improper user equivalence configuration or due to insufficient permissions.

Action: Verify that user equivalence is correctly configured and that the mentioned files have the correct permissions.

PRKC-1039: Error creating directories listed in {0} on node {1}

Cause: Might be due to an improper user equivalence configuration or due to insufficient permissions.

Action: Verify that user equivalence is correctly configured and that the mentioned files have the correct permissions.

PRKC-1040: Remote Shell is not known yet.

Cause: User equivalence is not yet configured.

Action: Configure user equivalence.

PRKC-1041: Remote Copy command is not known yet.

Cause: User equivalence is not yet configured.

Action: Configure user equivalence.

PRKC-1042: The Remote Shell {0} requested by client is not recognized

Cause: The system does not recognize the given remote shell command.

Action: Use ssh or rsh as the remote shell.

PRKC-1043: The Remote Copy command {0} requested by client is not recognized

Cause: The system does not recognize the given remote copy command.

Action: Use scp or rcp for remote copy.

PRKC-1044: Failed to check remote command execution setup for node {0} using shells {1} and {2}

Cause: Could not find the remote shell commands {1} and {2} on node {0}.

Action: Configure user equivalence. Use environment variables to pass remote shell commands.

PRKC-1045: Node name is null

Cause: Node name received is null.

Action: Pass non-null node name to method/command.

PRKC-1046: The file list passed to the method was null

Cause: The list of files obtained is null.

Action: Pass non-null file list to method.

PRKC-1047: The directory list passed to the method was null

Cause: The list of directories obtained is null.

Action: Pass non-null directory list to method.

PRKC-1048: {0} is not supported on Windows platform

Cause: Unsupported operation.

Action: Do not attempt {0} on Windows.

PRKC-1049: CRS is not supported in version {0}. It is supported from version {1} onwards

Cause: Unsupported operation.

Action: Upgrade to version {1} before attempting CRS commands.

PRKC-1050: EVM is not supported in version {0}. It is supported from version {1} onwards

Cause: Unsupported operation.

Action: Upgrade to version {1} before attempting to use EVM.

PRKC-1051: Invalid data to set for registry key {0}.

Cause: Data validation failed for registry operation.

Action: Check data passed as value for registry key {0}.

PRKC-1052: Invalid data type for registry key {0}.

Cause: Data validation failed for registry key {0}.

Action: Check data type of value for registry key {0}.

PRKC-1053: Error returned from node {0} is \"{1}\"

Cause:

Action: Contact your customer support representative.

PRKC-1054: Node {0} is not accessible

Cause: Might be due to a network issue or due to an improper user equivalence configuration.

Action: Contact your network or system administrator to verify connectivity and the health of node {0}.

PRKC-1055: Directory name passed was null

Cause:

Action: Ensure non-null directory name is passed to command.

PRKC-1056: Failed to get the hostname for node {0}

Cause: Unable to fetch the hostname.

Action: Check hostname of node {0}.

PRKC-1057: The computername and hostname do not match for node {0}. Make sure that they match.

Cause:

Action: Contact your network or computer administrator to reconfigure the computer name and/or the hostname before re-attempting the operation.

PRKC-1058: The hostname registry key {0} has an empty value on node {1}

Cause: Null value was passed for the registry key {0}.

Action: Contact your customer service representative.

PRKC-1059: The computername registry key {0} has an empty value on node {1}

Cause: Null value was passed for the registry key {0}.

Action: Contact your customer service representative.

PRKC-1060: Cluster misconfigured on node \"{0}\", \"{1}\" property in {2} does not match with configured data in Oracle Cluster Registry

Cause: Improper setup for the cluster.

Action: Contact your customer service representative.

PRKC-1061: Failed to list contents of the directory {0}

Cause: Might be due to insufficient permission.

Action: Check the permission and the contents of directory {0}.

PRKC-1062: Node list is null

Cause: The node list is null.

Action: Pass non-null node list to method/command.

PRKC-1063: Node list {0} contained {1} nodes, minimum number of nodes required is {2}

Cause:

Action: Modify nodelist to contain appropriate number of nodes before retrying method/command.

PRKC-1064: Node list {0} contained {1} nodes whereas the cluster has {2} nodes

Cause: The node list that is configured for the cluster does not match the given node list.

Action: Modify the nodelist to contain {2} nodes.

PRKC-1065: Nodes {0} do not belong to this cluster

Cause: The provided nodes are not configured for the cluster.

Action: Remove the invalid nodes that do not belong to cluster.

PRKC-1066: Failed to retrieve node number for node \"{0}

Cause: Cluster was not properly setup.

Action: Check CRS logs for detailed error information.

PRKC-1067: File name is null

Cause:

Action: Change file name to non-null value and retry method invocation.

PRKC-1068: Failed to get private interconnect node name for node {0}, {1}

Cause: Cluster was not properly setup.

Action: Check logs for detailed error information.

PRKC-1069: Failed to get sub keys of \"{0}\" registry key, {1}

Cause: Either the parent or the sub-key does not exist.

Action: Contact your customer service representative.

PRKC-1070: Failed to get sub keys of \"{0}\" registry key on node \"{1}\", {2}

Cause: Either the parent or the sub-key does not exist.

Action: Contact your customer service representative.

PRKC-1071: Nodes \"{0}\" did not respond to ping in \"{1}\" seconds, {2}

Cause: The node might be down or there could be a network outage.

Action: Contact your network administrator to verify connectivity between the specified nodes.

PRKC-1072: Invalid node \"{0}\" specified for checking the result of operation

Cause: Node name validation for the operation failed.

Action: Check the value of node {0}. Specify valid values for the nodes.

PRKC-1073: Failed to transfer directory \"{0}\" to any of the given nodes \"{1}\". {2}

Cause: Could not copy the directory \"{0}\" to any of the given nodes \"{1}\".

Action: Verify that user equivalence is properly configured. Check the log files for details.

PRKC-1074: Error on node {0}:{1}

Cause:

Action: Contact your customer support representative.

PRKC-1075: User equivalence does not exist with nodes \"{0}\", {1}

Cause: User equivalence was not configured.

Action: Contact your system administrator to verify user equivalence between the specified nodes.

PRKC-1076: Oracle home is null

Cause:

Action: Contact your customer support representative.

PRKC-1077: List file {0} has the following invalid files:{1}

Cause: Filename validation failed for the given files.

Action: Remove invalid entries from list file.

PRKC-1078: Failed to transfer directories listed in \"{0}\" to any of the given nodes \"{1}\": {2}

Cause: Could not copy the directory \"{0}\" to any of the given nodes \"{1}\".

Action: Verify that user equivalence is properly configured. Check the log files for details.

PRKC-1079: {0} is not contained within an Oracle home

Cause:

Action: Use {0} contained within an Oracle home.

PRKC-1080: Failed to transfer file \"{0}\" to any of the given nodes \"{1}\". {2}

Cause: Could not copy the file \"{0}\" to any of the given nodes \"{1}\".

Action: Verify that user equivalence is properly configured. Check the log files for details.

PRKC-1081: Failed to transfer listed files in \"{0}\" to any of the given nodes \"{1}\". {2}

Cause: Could not copy the files \"{0}\" to any of the given nodes \"{1}\".

Action: Verify that user equivalence is properly configured. Check the log files for details.

PRKC-1082: Failed to create listed directory in \"{0}\" to any of the given nodes \"{1}\". {2}

Cause: Could not create the directory \"{0}\" to any of the given nodes \"{1}\".

Action: Verify that user equivalence is properly configured. Verify the permissions for the parent directory on the given nodes.

PRKC-1083: Failed to remove listed directory in \"{0}\" to any of the given nodes \"{1}\". {2}

Cause: Could not remove the directory \"{0}\" to any of the given nodes \"{1}\".

Action: Verify that user equivalence is properly configured. Verify the permissions for the parent directory on the given nodes.

PRKC-1084: Failed to create path \"{0}\" in any of the given nodes \"{1}\". {2}

Cause: Could not create the path \"{0}\" to any of the given nodes \"{1}\".

Action: Verify that user equivalence is properly configured. Verify the permissions for the parent directory on the given nodes.

PRKC-1085: Failed to retrieve Oracle private name for node \"{0}\"

Cause: Cluster is not properly configured.

Action: Check whether node {0} is part of the cluster. If so, contact your customer support representative. If not, correct node name.

PRKC-1086: Unknown path type \"{0}\" specified for path \"{1}\" existence check

Cause: Unknown path type.

Action: Specify a valid path type for the path existence check.

PRKC-1087: Failed to retrieve virtual IP for node \"{0}\"

Cause: Cluster is not properly configured.

Action: Check whether node {0} is part of the cluster. If so, contact your customer support representative. If not, correct node name.

PRKC-1088: Failed to recursively list files from the directories specified in file: {0}

Cause: File access issue.

Action: Verify that the files exist and make sure that the files permissions are correct.

PRKC-1089: List file {0} is empty

Cause: An empty list file was passed to a transfer/copy command.

Action: Edit the contents of the list file before reattempting the command.

PRKC-1090: Failed to update environment on nodes \"{0}\", {1}

Cause: Might be due to improper privilege.

Action: Check user privilege on nodes \"{0}\".

PRKC-1091: The include list file passed is null

Cause:

Action: Pass non-null include file list to retry command/method.

PRKC-1092: Failed to retrieve the location of votedisks: {0}

Cause: Node name passed to command is not part of cluster.

Action: Invalid node name passed to command. Use a name for a node that is part of the cluster.

PRKC-1093: Failed to retrieve the version of crs software on node \"{0}\" : {1}

Cause: Cluster is not setup properly.

Action: Contact your customer support representative.

PRKC-1094: Failed to retrieve the active version of crs: {0}

Cause: Cluster is not setup properly.

Action: Contact your customer support representative.

PRKC-1095: OLE initialization or OCX load error while registering OCX library on nodes \"{0}\", {1}

Cause: Cluster is not setup properly.

Action: Contact your customer support representative.

PRKC-1096: Failed to do node reboot on nodes \"{0}\", {1}

Cause: Cluster is not setup properly.

Action: Contact your customer support representative.

PRKC-1097: PRKC- The file, \"{0}\" is not an OCX

Cause: Cluster is not setup properly.

Action: Contact your customer support representative.

PRKC-1098: Failed to create new listfile with proper UNC path from the path specified in file: {0}

Cause: Internal error.

Action: Contact your customer support representative.

PRKC-1099: The host names or IP addresses passed as an argument are null

Cause:

Action: Pass non-null arguments for hostnames or IP addresses.

PRKC-1100: The network port {0} is already in use

Cause: Preoccupied network port {0}.

Action: Free port {0} for use by CRS infrastructure.

PRKC-1101: The network port {0} is not available for use.

Cause: The port {0} might be used by another application.

Action: Make port {0} available for use.

PRKC-1102: CSS is not configured with local-only OCR on node \"{0}\"

Cause: CSS is not configured to run in a stand alone environment.

Action: Follow the documentation to setup CSS properly.

PRKC-1102: CSS is not configured with local-only OCR on node \"{0}\"

Cause: CSS is not configured to run in a standalone environment.

Action: Follow the documentation to setup CSS properly.

PRKC-1103: Failed to check CSS status for any of the given nodes \"{0}\", {1}

Cause: CSS might be down. User equivalence might not be set or there could be a network issue.

Action: Check the logs to see details.

PRKC-1104: Property, \"{0}\" is not found in file \"{1}\"

Cause: File \"{1}\" does not contain the property \"{0}\".

Action: Check the file '{1}'.

PRKC-1105: Failed to retrieve the local CSS home for node \"{0}\"

Cause: Cluster is not setup properly.

Action: Contact your customer support representative.

PRKC-1106: Failed to remove file \"{0}\" on node \"{1}\", {2}

Cause: Could not remove the file \"{0}\" on node \"{1}\".

Action: Verify that user equivalence is properly configured. Verify the permissions for the parent directory as well as for the file on the given nodes.

PRKC-1107: Failed to create symbolic link from file \"{0}\" to \"{1}\" on node \"{2}\", {3}

Cause: Could not create symbolic link.

Action: Verify that user equivalence is properly configured. Verify the permissions for the parent directory as well as for the file on the given nodes.

PRKC-1108: Failed to copy file \"{0}\" on node \"{1}\" to file \"{2}\" on node \"{3}\", {4}

Cause: Could not copy the file \"{0}\" on node \"{1}\".

Action: Verify that user equivalence is properly configured. Verify the permissions for the parent directory as well as for the file on the given nodes.

PRKC-1109: Failed to move file \"{0}\" to file \"{1}\" on node \"{2}\", {3}

Cause: Could not move the file \"{0}\" to file \"{1}\" on node \"{2}\".

Action: Verify that user equivalence is properly configured. Verify the permissions for the parent directory as well as for the file on the given nodes.

PRKC-1110: Failed to create directory \"{0}\" on node \"{1}\", {2}

Cause: Could not create the directory \"{0}\" on node \"{1}\".

Action: Verify that user equivalence is properly configured. Check the log files for details.

PRKC-1111: Failed to remove directory \"{0}\" on node \"{1}\", {2}

Cause: Could not remove the directory \"{0}\" on node \"{1}\".

Action: Verify that user equivalence is properly configured. Check the log files for details.

PRKC-1112: Failed to list contents of directory \"{0}\" on node \"{1}\", {2}

Cause: Could not read the directory \"{0}\" on node \"{1}\".

Action: Verify that user equivalence is properly configured. Check the log files for details.

PRKC-1113: Directory \"{0}\" does not exist on node \"{1}\"

Cause: The mentioned directory does not exist.

Action: Check for existence of directory {0}. Use absolute path for the directory.

PRKC-1114: Registry key name is null

Cause:

Action: Correct registry key name passed to command.

PRKC-1115: Registry subkey name is null

Cause:

Action: Correct registry subkey name passed to command.

PRKC-1116: Failed to create registry subkey \"{0}\" under key \"{1}\" on node \"{2}\", [3]

Cause: Registry operation failed.

Action: See logs for detail.

PRKC-1117: Failed to delete registry subkey \"{0}\" under key \"{1}\" on node \"{2}\", [3]

Cause: Registry operation failed.

Action: See logs for detail.

PRKC-1118: Failed to check existence of registry key \"{0}\" on node \"{1}\", [2]

Cause: Registry operation failed.

Action: See logs for detail.

PRKC-1119: Failed to set data for value \"{0}\" of registry key \"{1}\" on node \"{2}\", [3]

Cause: Registry operation failed.

Action: See logs for detail.

PRKC-1120: Failed to retrieve data for value \"{0}\" of registry key \"{1}\" on node \"{2}\", [3]

Cause: Registry operation failed.

Action: See logs for detail.

PRKC-1121: Failed to delete value \"{0}\" of registry key \"{1}\" on node \"{2}\", [3]

Cause: Registry operation failed.

Action: See logs for detail.

PRKC-1122: Service name is null

Cause: Empty service name provided.

Action: Reattempt the command with a non-null service name.

PRKC-1123: Failed to create service \"{0}\" on node \"{1}\", [2]

Cause:

Action: Please refer to logs for detailed error information.

PRKC-1124: Failed to create service dependency between services \"{0}\" and \"{1}\" on node \"{2}\", [3]

Cause:

Action: Please refer to logs for detailed error information.

PRKC-1125: Failed to start service \"{0}\" on node \"{1}\", [2]

Cause:

Action: Please refer to logs for detailed error information.

PRKC-1126: Failed to stop service \"{0}\" on node \"{1}\", [2]

Cause:

Action: Please refer to logs for detailed error information.

PRKC-1127:Failed to delete service \"{0}\" on node \"{1}\", [2]

Cause:

Action: Please refer to logs for detailed error information.

PRKC-1128: Invalid Interface type {0} is specified as an argument.

Cause:

Action: Correct interface type before retrying command.

PRKC-1129: Invalid IP address type {0} is specified as an argument.

Cause:

Action: Change IP address type before reattempting command.

PRKC-1130: Version argument is null.

Cause:

Action: Correct version argument and retry method invocation.

PRKC-1131: Failed to set user permissions on path {0} on nodes {1}, [{2}]

Cause: Might be due to improper user equivalence setup.

Action: Verify that user equivalence is correctly configured.

PRKC-1132: Failed to set administrator permissions on path {0} on nodes {1}, [{2}]

Cause: Might be due to improper user equivalence setup.

Action: Verify that user equivalence is correctly configured.

PRKC-1133: Names of services on which this service \"{0}\" is dependent on is null.

Cause:

Action: Invalid dependency list passed to method. Correct dependency list and retry command.

PRKC-1134: Failed to remove service dependency between services \"{0}\" and \"{1}\" on node \"{2}\", [3]

Cause:

Action: Refer to logs for detailed error information.

PRKC-1135: Registry key name {0} is not valid, it must be fully defined registry key path

Cause: Invalid registry key format.

Action: Provide valid registry key path.

PRKD—Global Services Daemon Messages

PRKD-3000: Failed to initialize and register with clusterware

Cause: This can occur if cluster synchronization services was not functioning properly.

Action: Check the state of your clusterware by running olsnodes from ORA_CRS_HOME/bin. It should list the nodes in the cluster.

PRKD-3001: Ready to receive client requests

Cause: Global Services Daemon is ready.

Action: No action needed at this time.

PRKE—Global Services Daemon Controller Utility Messages

PRKE-1008: Failed to get list of active nodes from clusterware

Cause: Not operating in cluster mode, failure to load srvm libraries, failure to initialize clusterware context.

Action: Verify the health of the Oracle Clusterware by running the "lsnodes" command to obtain a list of the active nodes in the cluster. Also verify the presence of the SRVM libraries in the load library path. If the active node list is correct and the SRVM libraries are in the load path, then contact your customer support representative.

PRKE-1009: Failed to start GSD on local node

Cause: GSD is already running on the local node, or GSD could not be started because of some other reason.

Action: Run "srvctl status" to check if a daemon is already running on the local node. If a daemon is not running on local node, please contact your customer support representative to check for the other reasons for failure.

PRKE-1010: Failed to stop GSD on local node

Cause: GSD is not running on the local node, or GSD could not shutdown gracefully.

Action: Run "srvctl status" to check if a GSD is running on the local node. If a GSD is running, please contact your customer support representative to check for other reasons for failure.

PRKE-1011: Failed to get status of GSD on local node

Cause: Failure to get the list of active nodes from the cluster, or failure to query the list of live daemons from the clusterware.

Action: Run "lsnodes" to check the health of the clusterware. If the clusterware is working fine, check for the presence of srvm libraries in the load library path. If the libraries are present, please contact your customer support representative for further assistance.

PRKH—Server Manager (SRVM) Messages

PRKH-1000: Unable to load the SRVM HAS shared library

Cause: The system cannot find, or load the srvm has shared library.

Action: Check your system's library load path. Verify that the library exists, and that the libraries it depends on are readable and are in the load path.

PRKH-1001: HASContext Internal Error

Cause: An unexpected internal error has occurred while attempting to communicate with CRS.

Action: Contact Oracle support.

PRKH-1002: Internal HASContext Error: JNI Native Call Failure

Cause: An unexpected internal error has occurred while executing native code from java.

Action: Contact Oracle support.

PRKH-1003: Failed to allocate memory in native layer:

Cause: The SRVM framework was unable to allocate memory.

Action: Your system is running low on memory. Check the memory and swap space, resolve the problem and retry.

PRKH-1004: Failed to execute remote join cluster alias {0} for nodes:

Cause: Unable to execute a join cluster alias operation on another node.

Action: Check the failed nodes. Verify that the system can execute remote commands on those nodes. Check the cluster alias and its related functionality.

PRKH-1005: Failed to execute remote leave cluster alias {0} for nodes:

Cause: Unable to execute a leave cluster alias operation on another node.

Action: Check the failed nodes. Verify that the system can execute remote commands on those nodes. Check the cluster alias and its related functionality.

PRKH-1006: The following nodes are either not active or not configured:

Cause: The system was unable to find a node requested.

Action: Please check the node list supplied and verify that the node is available and configured.

PRKH-1007: Exception Caused by:

Cause: The current exception was caused by another earlier exception.

Action: Examine all of the nested exceptions to determine the root cause of the error.

PRKH-1008: Internal HASContext Error: Argument {0} must be set.

Cause: Internal software error.

Action: Contact Oracle support.

PRKH-1009: CRS HOME must be defined in the environment or in the Oracle Cluster Registry

Cause: The system is unable to determine the ORA_CRS_HOME for this CRS installation.

Action: Check that the Oracle Cluster Registry is properly configured and available to this node.

PRKH-1010: Unable to communicate with CRS services

Cause: The system is unable to communicate with the Oracle Clusterware services.

Action: Check that all of the CRS daemons are running and are properly configured. Verify that the current program is attempting to communicate with the correct CRS daemons.

PRKH-1011: Process does not have sufficient privileges to perform the requested operation. {0}

Cause: The user running the current operation is not permitted to execute a given operation.

Action: Check other messages in the error stack to determine which operation failed. Verify that you are running the operation as the correct user.

PRKH-1012: Unable to find or resolve user {0}

Cause: The system was unable to find the user name supplied.

Action: Check the username and try again.

PRKH-1013: The oracle home {0} does not exist

Cause: The system was unable to find the oracle home supplied.

Action: Check the full path of the oracle home and try again.

PRKI—Cluster Pre-Install Messages

PRKI-2059 Unable to copy files to the nodes

Cause: Lack of permission to create destination file or copy file to the destination location.

Action: Verify permission to create the destination file. Also, please make sure that the destination file is not currently in use.

PRKI-2060 Unable to create service on the nodes

Cause: Lack of permission to create the service, or service already exists.

Action: Verify that you have the permission to create a service on the destination node or nodes. Also, verify that the service does not already exist.

PRKI-2061 Unable to start service on the nodes

Cause: Service is not created, service is already running, service marked for deletion, service could not be started properly.

Action: Try to start the service manually from the service control panel and check the Windows Error message that it gives.

PRKI-2061 Unable to start service on the nodes

Cause: This could be because of several possible problems: the service is not created, the service is already running, the service is marked for deletion, or the service could not be started properly.

Action: Try to start the service manually from the service control panel and check the Windows error message that it gives.

PRKI-2064 Unable to update registry entries on the nodes

Cause: Lack of permission to modify registry entries on the nodes, error while updating the registries.

Action: Verify permission to modify registry entries.

PRKI-2066 Unable to find an Oracle disk partition. Please exit from the wizard, create Oracle partitions and try again.

Cause: Absence of Oracle disk partition, failure to load OLM dll.

Action: Verify that OLM dll (oobjlib.dll) is present in the load path. Also, verify that the Oracle disk partitions are present by going to the Disk Administrator and looking for the partitions.

PRKI-2114 Cannot collect and verify hardware information for all the nodes. Press abort to exit the wizard or ignore to continue.

Cause: Failure to connect to the remote nodes while adding a node to a cluster. Failure to collect VIA information if VIA was chosen.

Action: If you are trying to add a node to an existing cluster, then attempt to map a drive letter to a drive on each of the remote nodes (using commands such as "net use"). Also, verify that the you have permission to start a temporary service on the remote nodes to collect VIA information, in case your system uses VIA.

PRKI-2116 Cannot connect to remote drive on node {0}

Cause: Failure to connect to remote node.

Action: Try to map a driver letter from the local node to the remote node using commands such as "net use."

PRKI-2119 Unable to delete files on {0}. Install may not succeed if files are not deleted

Cause: Files that you are trying to delete do not exist or are currently being used.

Action: If the files you are trying to delete do not exist, ignore this error. If the files you are trying to delete are currently being used, then stop the processes that are using the files and retry.

PRKI-2120 Unable to delete service {0} on {1}. Install may not succeed if services are not deleted

Cause: Service does not exist, or service could not be deleted.

Action: Check the service control panel to see if the service exists. If the service does not exist, ignore this error. If the service exists, try to delete it from the command line using utilities like "net" and "sc." If this fails, check the Windows error message returned by these commands.

PRKN—Server Manager (SRVM) System Library Messages

PRKN-1008: Unable to load the shared library "{0}" or a dependent library, from {1}={2} [{3}]

Cause: The SRVM framework was unable to load a shared library to perform native operations.

Action: Make sure that the library in question is installed and in a location where it can be loaded. Check the environment setting that determines shared library loading on your platform.

PRKN-1009: Native System Internal Error

Cause: An unexpected internal error occurred while attempting to execute a native operation from Java.

Action: Contact Oracle Support.

PRKN-1010: This system is not properly configured as a cluster

Cause: SRVM was unable to find the system libraries which are required in order to use the system as a cluster.

Action: Verify that the Oracle Clusterware installation succeeded and attempt to execute the 'srvctl status' command to obtain more diagnostic information.

PRKN-1011: Failed to retrieve value for "{0}" under registry key "{1}" on node "{2}", {3}

Cause: There was a problem accessing the Windows registry key.

Action: Open the Windows Registry editor on the node and verify that the Registry key exists and that it contains the subkey.

PRKO—Server Control (SRVCTL) Utility Messages

PRKO-2001: "Invalid command line syntax"

Cause: An invalid SRVCTL command line was entered.

Action: Use -h SRVCTL command line option to find out the correct command line syntax and re-enter the command.

PRKO-2002: "Invalid command line option: "

Cause: An invalid SRVCTL command line option was entered.

Action: Use -h SRVCTL command line option to find out the correct command line syntax and re-enter the command.

PRKO-2003: "Invalid command line option value: "

Cause: An invalid SRVCTL command line option value was entered.

Action: Use -h SRVCTL command line option to find out the correct command line syntax and re-enter the command.

PRKO-2004: "Repetition of command line option: "

Cause: Duplicate SRVCTL command line option was entered.

Action: Eliminate the duplicate and re-enter the command.

PRKO-2005: "Application error: Failure in getting Cluster Database Configuration for: "

Cause: An error occurred when getting Cluster Database configuration for the named database.

Action: Make sure that the database has been configured in Cluster Database Configuration Repository; make sure that GSDs are running on each node in the cluster.

PRKO-2005: "Application error: Failure in getting Cluster Database Configuration for: "

Cause: An error occurred when getting Cluster Database configuration for the named database.

Action: Verify that the database has been configured in the Oracle Cluster Registry; make sure that the GSDs are running on each node in the cluster.

PRKO-2006: "Invalid node name: "

Cause: An invalid node name was entered.

Action: Use the correct node name. A valid node name should match the output from 'lsnodes' and must not contain domain name.

PRKO-2007: "Invalid instance name: "

Cause: An invalid instance name was entered.

Action: Use the correct instance name for the database. Run the 'srvctl config database -d <db_name>' command to identify all instances of the database in the Oracle Cluster Registry.

PRKO-2008: "Invalid connect string: "

Cause: An invalid connect string was entered.

Action: Use the correct connect string syntax: *<user>*/*<password>*[as *<role>*].

PRKO-2009: "Invalid name/value string: "

Cause: An invalid environment name/value pair was entered during SRVCTL setenv command.

Action: Make sure that the correct name/value string format is used: *<name>*=*<value>*.

PRKO-2010: "Error in adding instance to node: "

Cause: An error occurred when adding instance to the Oracle Cluster Registry.

Action: Use the 'srvctl config database -d <db_name>' command to check if the database has been configured in the Oracle Cluster Registry; make sure that the GSDs are running on each node in the cluster.

PRKO-2011: "Error in removing instance: "

Cause: An error occurred when removing an instance from the Oracle Cluster Registry.

Action: Use the 'srvctl config database -d <db_name>' command to check if the database and instance have been configured in the Oracle Cluster Registry; make sure that the GSDs are running on each node in the cluster.

PRKO-2012: "Error in moving instance to node: "

Cause: An error occurred when changing the instance and node mapping in the Oracle Cluster Registry.

Action: Use the 'srvctl config database -d <db_name>' command to check if the database and instance have been configured in the Oracle Cluster Registry; make sure that the GSDs are running on each node in the cluster.

PRKO-2013: "Error in setting env: "

Cause: An error occurred when setting environment variables for a database or an instance.

Action: Use 'srvctl config database -d <db_name>' command to check if the database and/or the instance have been configured in the Oracle Cluster Registry; make sure that the GSDs are running on each node in the cluster.

PRKO-2014: "Error in unsetting env: "

Cause: An error occurred when unsetting environment variables for a database or an instance.

Action: Use the 'srvctl getenv' command to check if the environment variable exists for the database or instance; make sure that the database and/or the instance have been configured in the Oracle Cluster Registry; make sure that the GSDs are running on each node in the cluster.

PRKO-2015: "Error in checking condition of instance on node: "

Cause: Could not get status information of an instance.

Action: Use the 'srvctl config database -d <db_name>' command to check if the instance has been configured in the Oracle Cluster Registry; make sure that the GSDs are running on each node in the cluster.

PRKO-2016: "Error in checking condition of listener on node: "

Cause: Could not get status information of a listener.

Action: Check if the listener has been configured in the listener configuration file and if the database instance on the node is listed in SID_NAME in the listener configuration; make sure that GSDs are running on each node in the cluster.

PRKO-2017: "Service {0} is not supported on instance {1}."

Cause: The named instance is not configured in the service configuration.

Action: In service related operations, make sure that you operate on the instances that are configured for the service.

PRKO-2019: Cannot change management policy when database is disabled. Either remove -y option or enable database before before running this command

Cause: The database resource attributes cannot be changed when it is in online state.

Action: Stop the database using the 'srvctl stop database' command and then re-run the command to change the policy.

PRKO-2101: "Instance is disabled: "

Cause: The instance is disabled and cannot be started.

Action: Enable the instance before starting.

PRKO-2102: "Service {0} is not supported on instance {1}."

Cause: The named instance is not configured in the service configuration.

Action: In service related operations, make sure you operate on the instances that are configured for the service.

PRKO-2104: "Error in checking condition of service: "

Cause: There is a problem with the CRS daemon, or the service resource is in an unknown state.

Action: Check whether the daemon is up and listening to the port that SRVCTL is communicating with. Check the service resource status.

PRKO-2105: "Error in checking condition of VIP on node: "

Cause: There is a problem with the CRS daemon, or the VIP resource is in an unknown state.

Action: Check whether the daemon is up and listening to the port that SRVCTL is communicating with. Check the VIP resource status.

PRKO-2106: "Error in checking condition of GSD on node: "

Cause: There is a problem with the CRS daemon, or the VIP resource is in an unknown state.

Action: Check whether the daemon is up and listening to the port that SRVCTL is communicating with. Check the VIP resource status.

PRKO-2108: "Node applications are still running on node: "

Cause: Node applications are running on the node.

Action: Stop the node applications before removing them.

PRKO-2109: "Invalid address string: "

Cause: Invalid address string.

Action: Use the correct format for address string.

PRKO-2112: "Some or all node applications are not removed successfully on node: "

Cause: There was a problem when removing the node applications.

Action: Details of the problem are given in the text.

PRKO-2113: "Instance {0} is already a preferred instance for service {1}."

Cause: The instance is configured as a preferred instance for the service.

Action: Pick another instance to configure.

PRKO-2114: "Instance {0} is already an available instance for service {1}."

Cause: The instance is configured as an available instance for the service.

Action: Select another instance to configure.

PRKO-2115: "Cannot remove node-level applications on node {0}, because a listener application exists. Remove the listener application first, then re-run this command."

Cause: A listener application exists.

Action: Remove the listener application first, then re-run this command.

PRKO-2116: "Error in checking condition of ONS daemon on node: "

Cause: There is a problem with the CRS daemon, or the ONS resource is in an unknown state.

Action: Check if the daemon is up and listening to the port that SRVCTL is communicating with. Check the ONS resource status.

PRKO-2117: "This command should be executed as the system privilege user."

Cause: User did not login as the system privilege user.

Action: Login as system privilege user and re-try.

PRKO-2118: "Error in checking condition of ASM instance {0} on node {1}."

Cause: There is a problem with the CRS daemon, or the ASM resource is in an unknown state.

Action: Check if the daemon is up and listening to the port that SRVCTL is communicating with. Check the ASM resource status.

PRKO-2119: "Error in checking enable/disable status of ASM instance {0} on node {1}."

Cause: There was some problem when checking the status of the ASM instance.

Action: Make sure that the Oracle Cluster Registry is functioning and that the CRS daemon is running.

PRKO-2120: "The internal database service {0} cannot be managed with srvctl."

Cause: The internal database service is not supposed to be a highly available service.

Action: Do not operate on the internal database service.

PRKP—Cluster Database Management Messages

PRKP-1000 "Cannot retrive configuration for cluster database {0}"

Cause: The cluster database configuration cannot be retrieved from the repository. This can occur either because the database was never registered, or because the repository itself has not been created.

Action: Check if the database has been configured by printing a list of all cluster databases using 'srvctl config'. If the repository has not been created, use 'srvconfig -init' to create it.

PRKP-1001 "Error starting instance {0} on node {1}"

Cause: The instance could not be started using the SQL*Plus startup command.

Action: Try starting the named instance manually using SQL*Plus to see why it failed.

PRKP-1002 "Error stopping instance {0} on node {1}"

Cause: The SQL*Plus shutdown command returned an error while stopping the instance.

Action: Try stopping the named instance manually using SQL*Plus to see why it failed.

PRKP-1003 "Startup operation partially failed"

Cause: Some components of the cluster database could not be started.

Action: See earlier error message for details.

PRKP-1004 "Shutdown operation partially failed"

Cause: Some components of the cluster database reported errors while being stopped.

Action: See earlier error messages for details.

PRKP-1005 "Failed to start up cluster database {0}"

Cause: The cluster database could not be started.

Action: See earlier error messages for details.

PRKP-1006 "Failed to shut down cluster database {0}"

Cause: The cluster database reported errors while being shut down.

Action: See earlier error messages for details.

PRKP-1007 "Failed to start all the listeners associated with all the instances of cluster database {0}"

Cause:

Action: Contact your customer support representative.

PRKP-1008 "Failed to start listeners associated with instance {0} on node {1}"

Cause:

Action: Contact your customer support representative.

PRKP-1009 "Failed to stop all the listeners associated with all the instances of cluster database {0}"

Cause: Either the listener name associated with an instance could not be determined, or "lsnrctl stop" failed for a listener.

Action: Verify that listener.ora contains a SID_LIST entry for each instance of the named database, and that the lsnrctl stop command succeeds for those listeners.

PRKP-1010 "Failed to stop all the listeners associated with instance {0} on node{1}"

Cause: Either the listener name associated with an instance could not be determined, or "lsnrctl stop" failed for a listener.

Action: Verify that listener.ora contains a SID_LIST entry for each instance of the named database, and that the lsnrctl stop command succeeds for those listeners.

PRKP-1011 "Failed to get all the listeners associated with instance {0} on node{1}"

Cause: The listener name associated with an instance could not be determined.

Action: Ensure that listener.ora contains a SID_LIST entry for the named instance.

PRKP-1012 "Invalid environment variable {0} setting for cluster database {1}"

Cause: The argument to the -t option is not of the form <name>=<value> or it contains special characters.

Action: Ensure that the -t option has an argument of the form <name>=<value>. Enclose the argument to the -t flag in quotes.

PRKP-1013 "{0}: undefined environment variable for cluster database {1}"

Cause: The named environment variable is not defined for the named cluster database.

Action: Set a value for the variable with "srvctl set env."

PRKP-1014 "{0}: undefined environment variable for instance {1} of cluster database {2}"

Cause: The named environment variable is not defined for the given instance.

Action: Set a value for the variable with "srvctl set env."

PRKP-1015 "{0}: undefined environment variable"

Cause: The named environment variable is not defined.

Action: Set a value for the named environment variable with "srvctl set env."

PRKP-1016 "Database {0} already enabled"

Cause: An attempt was made to enable a database that is already enabled.

Action: No action required.

PRKP-1017 "Instance {0} already enabled."

Cause: An attempt was made to enable an instance that is already enabled.

Action: No action required.

PRKP-1018 "Service {0} already enabled."

Cause: An attempt was made to enable a service that is already enabled.

Action: No action required.

PRKP-1019 "Database {0} already disabled."

Cause: An attempt was made to disable a database that is already disabled.

Action: No action required.

PRKP-1020 "Instance {0} already disabled."

Cause: An attempt was made to disable an instance that is already disabled.

Action: No action required.

PRKP-1021 "Service {0} already disabled."

Cause: An attempt was made to disable a service that is already disabled.

Action: No action required.

PRKP-1022 "The database {0} is still running."

Cause: An attempt was made to delete a database that is still running.

Action: Stop the database using 'srvctl stop database' before deleting the database.

PRKP-1023 "The instance {0} is still running."

Cause: An attempt was made to delete an instance that is still running.

Action: Stop the instance using 'srvctl stop instance' before deleting the instance.

PRKP-1024 "The service {0} is still running."

Cause: An attempt was made to delete a service that is still running.

Action: Stop the service using 'srvctl stop service' before deleting the service.

PRKP-1025 "The service {0} does not exist."

Cause: An attempt was made to operate on a non-configured service.

Action: Check if the service is configured through 'srvctl status service'.

PRKP-1026 "No instance found for database {0}."

Cause: An attempt was made to operate on a non-configured instance.

Action:

PRKP-1027 "Instance {0} is not found for database {1}."

Cause: An attempt was made to operate on a non-configured instance.

Action: Check if the instance is configured through 'srvctl config instance'.

PRKP-1028 "No preferred instance(s) for service {0}."

Cause: An attempt was made to create a service without preferred instances.

Action: Supply preferred instances for the service through 'srvctl create service'.

PRKP-1029 "Failed to register the service {0}."

Cause: Internal error.

Action: Contact support.

PRKP-1030 "Failed to start the service {0}."

Cause: Internal error.

Action: Contact support.

PRKP-1031 "Failed to stop the service {0}."

Cause: Internal error.

Action: Contact support.

PRKP-1032 "Cannot start the disabled service {0}."

Cause: Internal error.

Action: Contact support.

PRKP-1033 "Cannot relocate service {0} from instance {1} to instance {2}."

Cause: Internal error.

Action: Contact support.

PRKP-1034 "{0}: undefined environment variable for node {1}."

Cause: Internal error.

Action: Contact support.

PRKP-1035 "Invalid environment variable {0} setting for node {1}."

Cause:

Action: Contact support.

PRKP-1036 "Failed to unregister HA resource {0}."

Cause:

Action: Contact support.

PRKP-1037 "Failed to create cluster database {0}."

Cause:

Action: Contact support.

PRKP-1038 "Invalid instance {0} specified for the Service {1}."

Cause:

Action: Contact support.

PRKP-1039 "operation result is null"

Cause:

Action: Contact support.

PRKP-1041: "Failed to reload all the listeners associated with cluster database {0}"

Cause:

Action: Refer to CRS logs for detailed error information.

PRKP-1042: "Failed to reload all the listeners associated with instance {0} on node{1}"

Cause:

Action: Refer to CRS logs for detailed error information.

PRKP-1043: "Failed to reload listeners on node {0}"

Cause:

Action: Refer to CRS logs for detailed error information.

PRKP-1044 "Failed to enable the database {0}."

Cause:

Action: Contact support.

PRKP-1045 "Failed to disable the database {0}."

Cause:

Action: Contact support.

PRKP-1046 "Failed to enable the instance {0}."

Cause:

Action: Contact support.

PRKP-1048 "Failed to change configuration for service {0}."

Cause:

Action: Contact support.

PRKP-1049 "{0}: undefined environment variable for service {1} of cluster database {2}"

Cause: The environment variable is not defined for the service.

Action: No action required.

PRKP-1050 "Failed to remove the service {0}."

Cause: There was a problem while executing the crs_unregister command.

Action: Verify whether the crs_unregister command succeeds in unregistering a CRS resource.

PRKP-1051 "Failed to remove the service {0} on instance {1}."

Cause: There was a problem while executing the crs_unregister command.

Action: Verify whether the crs_unregister command succeeds in unregistering a CRS resource.

PRKP-1052 "Failed to enable the service {0}."

Cause: There was a problem while executing the crs_register command.

Action: Verify whether the crs_register command succeeds in registering a CRS resource.

PRKP-1053 "Failed to disable the service {0}."

Cause: There was a problem while executing the crs_register command.

Action: Verify whether the crs_register command succeeds in registering a CRS resource.

PRKP-1054 "Failed to enable the service {0} on instance {1}."

Cause: There was a problem while executing the crs_register command.

Action: Verify whether the crs_register command succeeds in registering a CRS resource.

PRKP-1055 "Failed to disable the service {0} on instance {1}."

Cause: There was a problem while executing the crs_register command.

Action: Verify whether the crs_register command succeeds in registering a CRS resource.

PRKP-1056 "Failed to get the status of the resource {0}."

Cause: There was a problem while executing the crs_stat command.

Action: Verify whether the crs_stat command gives the status of the CRS resources registered.

PRKP-1057 "Failed to set the environment for service {0}."

Cause: There was a problem while accessing the OCR.

Action: Check whether the OCR is accessible by executing a srvctl config command.

PRKP-1058 "Failed to unset the environment for service {0}."

Cause: There was a problem while accessing the OCR.

Action: Check whether the OCR is accessible by executing the srvctl config command.

PRKP-1059 "Failed to get the environment for service {0}."

Cause: There was a problem while accessing the OCR.

Action: Check whether the OCR is accessible by executing the srvctl config command.

PRKP-1060 "Failed to get CRS home."

Cause: Internal error.

Action: Contact support.

PRKP-1061 "Failed to modify the database {0}. "

Cause: Internal error.

Action: Contact support.

PRKP-1062 "Service {0} is already running."

Cause: Attempted to start a service that is already running.

Action: None required.

PRKP-1063 "Service {0} is already stopped."

Cause: Attempted to stop a service that is already stopped.

Action: None required.

PRKP-1064 "Service {0} is already running on instance {1}."

Cause: Attempted to start a service on an instance where it is already running.

Action: None required.

PRKP-1065 "Service {0} is already stopped on instance {1}."

Cause: Attempted to stop a service on an instance where it is already stopped.

Action: None required.

PRKP-1066 "Instance {0} is not an available instance for service {1}."

Cause:

Action: Contact support.

PRKP-1067 "Instance {0} is the last available instance for service {1}. Try modify service instead."

Cause:

Action: Contact support.

PRKP-1068 "Cannot stop the critical instance {0} in critical standby database {1} because it would result in a shutdown of the primary database."

Cause: Attempted to stop the critical instance in a critical standby database while the primary database is running.

Action: Do not stop the critical instance in a critical standby database while the primary database is running.

PRKP-1069 "Failed to change domain of the database {0} to {1}, because this domain name is already used by service {2} configured under the database."

Cause: Attempted to change the database domain when there are services configured with this domain.

Action: Do not change the database domain when there are services configured with that domain.

PRKP-1070 "Service name {0} contains illegal characters."

Cause: Invalid characters have been specified in the service name given.

Action: Supply a name for the service with the character set [a-zA-Z0-9_.].

PRKP-1071: "Database unique name or instance name {0} contains illegal characters {1}."},

Cause:

Action: Correct db/instance name before retrying method/command.

PRKP-1072 "Failed to create service {0} for database {1}, because the specified service domain name is the same as the database domain {2}."

Cause:

Action:

PRKP-1073 "Cannot create database {0} because a database named {1} already exists."

Cause: Attempted to create a database that already exists.

Action: Choose a different name for the database being created.

PRKP-1074 "Failed to relocate a service resource to instance {0} during modifying service configuration for service {1}."

Cause: Internal error.

Action: Contact support.

PRKP-1075 "Instance {0} is the last preferred instance for service {1}."

Cause: Internal error.

Action: Contact support

PRKP-1076: Instance name passed to the method is null

Cause: The Oracle instance SID that you are passing to this API cannot be null.

Action: Make sure that the Oracle instance is configure before this call. Check client specific error message for details.

PRKP-1077: Oracle home is null for the cluster database {0}

Cause: Oracle home specified cannot be null.

Action: Make sure Oracle home is properly set. Check client specific error message for more details.

PRKP-1078: Failed to retrieve \"{0}\" attribute value from \"{1}\" file, {2}"

Cause: An error occurred while retrieving CRS attribute value because of some CRS/OCR error.

Action: Refer to CRS or OCR logs in CRS home/log/<hostname> for detailed error information.

PRKP-1079: Cannot start service {0} on disabled database {1}

Cause: Service cannot be started on a database which is already disabled.

Action: Enable the database using 'srvctl enable database' before re-attempting command.

PRKP-1080: "Cannot start service {0} on disabled instance {1}."

Cause: Service cannot be started in an instance which is already disabled.

Action: Enable the instance using 'srvctl enable instance' before attempting to start the service.

PRKP-1081: Database name passed to the method is null

Cause: The database name cannot be null to perform the operation.

Action: Verify the database name that you are using and retry the command

PRKP-1082: Instance \"{0}\" does not exist in database \"{1}\"

Cause: The configuration information for this instance may not exist in the OCR, or the instance was never configured.

Action: Verify the instance name that you are using before retrying the command. Alternatively, verify the instances that are configured with a database by running the command 'srvctl config database -d <database>'.

PRKP-1083: The service {0} already exists.

Cause: You cannot have multiple database services with same name.

Action: Make sure you use unique service names for each database.

PRKP-1084: Database \"{0}\" cannot have more than \"{1}\" services

Cause: The database supports only 115 user services. You cannot configure more than 115 services.

Action: If you must add this service, then remove some of the unused services using the command 'srvctl removes services' before adding any new services.

PRKR—Cluster Registry Messages

PRKR-1001 "cluster database {0} does not exist"

Cause: The cluster database was never configured in the OCR.

Action: Check if the database has been configured by printing a list of all cluster databases using 'srvctl config'.

PRKR-1002 "cluster database {0} already exists"

Cause: An attempt was made to configure a cluster database that already exists in the OCR.

Action: Check if the database has already been configured by printing a list of all cluster databases using 'srvctl config'.

PRKR-1003 "instance {0} does not exist"

Cause: The named instance is not configured in the OCR.

Action: Use srvctl options to check if the instance was configured in the OCR.

PRKR-1004 "instance {0} already exists"

Cause: The named instance is already configured in the OCR.

Action: Use srvctl options to check if the instance has already been configured in the OCR.

PRKR-1005 "adding of cluster database {0} configuration failed, {1}"

Cause: An error occurred while attempting to add the cluster database configuration information to the OCR.

Action: Verify whether the OCR is accessible by using an 'ocrcheck' or 'srvctl config' command.

PRKR-1006 "deleting of cluster database {0} configuration failed, {1}"

Cause: Same as PRKR-1005.

Action: See earlier error messages.

PRKR-1007 "getting of cluster database {0} configuration failed, {1}"

Cause: Same as PRKR-1005.

Action: See earlier error messages.

PRKR-1008 "adding of instance {0} on node {1} to cluster database {2} failed, {3}"

Cause: Same as PRKR-1005.

Action: See earlier error messages.

PRKR-1009 "deleting of instance {0} from cluster database {1} failed, {2}"

Cause: Same as PRKR-1005.

Action: See earlier error messages.

PRKR-1010 "moving of instance {0} to node {1} of cluster database {2} failed, {3}"

Cause: Same as PRKR-1005.

Action: See earlier error messages.

PRKR-1011 "renaming of instance {0} to instance {1} of cluster database {2} failed, {3}"

Cause: Same as PRKR-1005.

Action: See earlier error messages.

PRKR-1016 "reading of cluster database {0} configuration failed, {1}, {2}"

Cause: Same as PRKR-1005.

Action: See earlier error messages.

PRKR-1017 "writing of cluster database {0} configuration failed, {1}, {2}"

Cause: Same as PRKR-1005.

Action: See earlier error messages.

PRKR-1018 "reading of directory failed, {0}, {1}"

Cause: Internal error.

Action: Contact your customer support representative.

PRKR-1019 "writing of directory failed, {0}, {1}"

Cause: Internal error.

Action: Contact your customer support representative.

PRKR-1020 "reading of version information failed, {0}, {1}"

Cause: Same as PRKR-1005.

Action: See earlier error messages.

PRKR-1021 "writing of version information failed, {0}, {1}"

Cause: Same as PRKR-1005.

Action: See earlier error messages.

PRKR-1022 "raw device {0} contains incompatible version, {1} != {2}"

Cause: An attempt was made to use an incompatible version of the OCR.

Action: Contact your customer support representative.

PRKR-1023 "file {0} does not exist"

Cause: The named file did not exist.

Action: Check if the file exists.

PRKR-1024 "file {0} does not have {1} permissions"

Cause: The named file did not have the specified permission.

Action: Try changing the permission on the file to the specified permission.

PRKR-1025 "file {0} does not contain property {1}"

Cause: The file did not contain the specified property.

Action: Contact your customer support representative.

PRKR-1026 "property {0} not set in file {1}"

Cause: The file did not contain the specified property.

Action: Contact your customer support representative.

PRKR-1027 "failed to retrieve list of cluster databases"

Cause: Same as PRKR-1005.

Action: See earlier error messages.

PRKR-1028 "raw device {0} is invalid\n[HINT: initialize raw device by using \"srvconfig\" tool]"

Cause: The cluster registry was never initialized.

Action: Use srvconfig -init to initialize the cluster registry.

PRKR-1038 "invalid argument {0} specified to -init option"

Cause:

Action:

PRKR-1039 "invalid option {0} specified"

Cause: The specified option was invalid.

Action: Check usage.

PRKR-1040 "missing <file> argument for {0} option"

Cause: The specified option was invalid for srvconfig.

Action: Check usage for details.

PRKR-1045 "raw device version \"{0}\""

Cause: Attempted to retrieve the version of the cluster registry.

Action: No action is required.

PRKR-1046 "srvconfig detected valid raw device \"{0}\"\n[HINT: please specify -init -f option to forcefully initialize it]"

Cause: A valid cluster registry was detected.

Action: No action is required.

PRKR-1047 "raw device {0} is in use by daemon(s) on node(s) {1}"

Cause: An attempt was made to initialize the OCR while the Global Services Daemons were up on one or more nodes in the cluster.

Action: Stop all of the Global Services Daemons on all of the nodes in the cluster by running the 'srvctl stop' command on every node. Try the 'srvconfig -init' operation again.

PRKR-1050 "file {0} creation in {1} directory failed, check permissions, etc."

Cause: Attempted to create a file in a directory which did not exist or which d id not have the right permissions.

Action: Create the directory if it did not exist or change the permission of the directory.

PRKR-1051 "file {0} does not contain an entry for dbname {1}"

Cause: Internal error.

Action: Contact your customer support representative.

PRKR-1052 "file name {0} is not of <cluster database name>.conf form"

Cause: An attempt was made to register a cluster database in the OCR and the file argument passed was not of the <cluster database name>.conf form.

Action: Refer to the usage of the srvconfig command for more information.

PRKR-1053 "invalid range {0} specified in node_list = {1}"

Cause:

Action: Contact your customer support representative.

PRKR-1054 "invalid parameter {0} specified in inst_oracle_sid = {1}"

Cause: Extra number of arguments provided to srvconfig.

Action: See usage of srvconfig for details.

PRKR-1055 "invalid extra arguments {0} specified to {1} option"

Cause: Provided invalid arguments to srvconfig.

Action: See usage of srvconfig for details.

PRKR-1056 "invalid registry entry {0} found, should be of form {1}"

Cause: Detected an invalid registry entry while attempting to add a 8.1.7 or earlier version of Oracle Parallel Server in the OCR.

Action: Contact your customer support representative.

PRKR-1057 "environment variable does not exist"

Cause: Attempted to retrieve non existing environment variable.

Action: Set the environment variable.

PRKR-1058 "Service {0} does not exist in cluster database {1}."

Cause: The named service is not configured in the OCR.

Action: Use srvctl options to check whether the service was configured in the OCR.

PRKR-1059 "Node {0} does not exist."

Cause: Node applications for the named node are not configured in the OCR.

Action: Use srvctl options to check whether the node applications were configured for the given node in the OCR.

PRKR-1060 "Failed to add configuration for node {0}"

Cause:

Action: Contact your customer support representative.

PRKR-1061 "Failed to run remote command to get node configuration for node {0}"

Cause: An internal error occurred while retrieving the configuration for the given node.

Action: Contact your customer support representative.

PRKR-1062 "Failed to find configuration for node {0}"

Cause:

Action: Contact your customer support representative.

PRKR-1063 "VIP {0} is already existing"

Cause:

Action: Contact your customer support representative.

PRKR-1064 "SRVM configuration operation failed due to Oracle Cluster Registry error :"

Cause: Error occurred while accessing the Oracle Cluster Registry.

Action: Contact your customer support representative.

PRKR-1066 "cluster database domain does not match"

Cause:

Action: Contact your customer support representative.

PRKR-1067 "Failed to get environment for cluster database {0}, {1}"

Cause: Unable to retrieve the environment configuration for the given cluster database from the OCR.

Action: Use the ocrdump utility to check whether the environment was configured for the given cluster database in the OCR.

PRKR-1068 "Failed to get environment for instance {1} of cluster database {0}, {2}"

Cause: Unable to retrieve the environment configuration for the given instance from the OCR.

Action: Use ocrdump to check if environment was configured for the given instance in the OCR.

PRKR-1069 "Failed to set environment for cluster database {0}, {1}"

Cause: Unable to update the environment configuration for the given cluster database in the OCR.

Action: Contact your customer support representative.

PRKR-1070 "Failed to set environment for instance {1} of cluster database {0}, {2}"

Cause: Unable to update the environment configuration for the given instance to the OCR.

Action: Contact your customer support representative.

PRKR-1071 "Failed to unset environment for cluster database {0}, {1}"

Cause: Unable to update the environment configuration for the given cluster database in the OCR.

Action: Contact your customer support representative.

PRKR-1072 "Failed to unset environment for instance {1} of cluster database {0}, {2}"

Cause: Unable to update the environment configuration for the given instance in the OCR.

Action: Contact your customer support representative.

PRKR-1073 "\n########## Configuration of nodeapps follows ##########\n"

Cause:

Action: Contact your customer support representative.

PRKR-1074 "\n########## Configuration of vip_range follows ##########\n"

Cause:

Action: Contact your customer support representative.

PRKR-1075 "Insufficient privileges for doing this operation"

Cause: User did not have sufficient privileges when running this command.

Action: Execute this command as a privileged user.

PRKR-1076 "This command cannot run when the Oracle RAC daemons (crsd, evmd, ocssd) are running. Make sure the daemons are not running before invoking this command"

Cause: The Oracle RAC daemons were running when this command was invoked.

Action: Make sure that the Oracle RAC daemons have been stopped before running this command.

PRKR-1077 "One or more arguments passed to the function are not valid"

Cause: One or more invalid arguments were passed to the given method.

Action: Contact your customer support representative.

PRKR-1078 "Database {0} cannot be administered using current version of srvctl. Instead run srvctl from {1}"

Cause: Using incorrect SRVCTL version.

Action: Contact your customer support representative.

PRKR-1079 "Failed to initialize the Oracle Cluster Registry"

Cause: Failed to initialize the Oracle Cluster Registry.

Action: Contact your customer support representative.

PRKS—Automatic Storage Management Messages

PRKS-1000: "ASM instance "{0}" already exists on node "{1}"

Cause: An attempt was made to add configuration information for an ASM instance on the node where it already exists.

Action: Check whether the ASM instance was configured on the node using 'srvctl config asm -n <node>' before adding the configuration for it.

PRKS-1001: ASM instance "{0}" does not exist on node "{1}"

Cause: The configuration for the ASM instance does not exist on the node.

Action: Check whether the ASM instance was configured on the node using 'srvctl config asm -n <node>' before performing the operation.

PRKS-1002: Failed to create CRS profile for ASM instance "{0}" on node "{1}", [{2}]

Cause: 'crs_stat -p' command failed for the ASM instance resource on the node.

Action: Contact Oracle Support.

PRKS-1003: Failed to register CRS resource for ASM instance "{0}" on node "{1}", [{2}]

Cause: crs_register command failed for ASM instance resource on the node.

Action: Contact Oracle Support.

PRKS-1004: Failed to unregister CRS resource for ASM instance "{0}" on node "{1}", [{2}]" }

Cause: A crs_unregister command failed for the ASM instance resource on the node.

Action: Check if there is a database instance that is dependent upon the ASM instance on the node by running the command crs_stat -p ora.<db>.<inst>.inst and see if ASM instance resource name appears in the required resources for the database instance. Remove the database instance's required resource dependency on the ASM instance by running a 'srvctl modify asm' command before retrying this operation.

PRKS-1005: Failed to create CRS resource for ASM instance "{0}" on node "{1}", [{2}]

Cause: crs_register command failed for the ASM instance resource on the node.

Action: Contact Oracle Support.

PRKS-1006: ASM instance "{0}" is already running on node "{1}".

Cause: An attempt was made to start a running ASM instance on the node.

Action: None.

PRKS-1007: ASM instance "{0}" is still running on node "{1}".

Cause: An attempt was made to remove the configuration for a running ASM instance on the node.

Action: Stop the ASM instance using 'srvctl stop asm -n <node> -i <inst>' command before performing the remove operation.

PRKS-1008: ASM instance "{0}" is not running on node "{1}".

Cause: An attempt was made to stop a non-running ASM instance on the node.

Action: None.

PRKS-1009: Failed to start ASM instance "{0}" on node "{1}", [{2}]

Cause: crs_start failed for the ASM instance resource on the node, may be due to invalid startup credentials or missing parameter file.

Action: Check if VIP resource is online on the node and then try to startup the ASM instance using SQL*Plus to get more diagnostic information.

PRKS-1010: Failed to start ASM instances "{0}" on node "{1}", [{2}]

Cause: See above.

Action: See above.

PRKS-1011: Failed to check status of ASM instance "{0}" on node "{1}", [{2}]

Cause: crs_stat failed for the ASM instance resource on the node.

Action: Contact Oracle Support.

PRKS-1012: Failed to stop ASM instance "{0}" on node "{1}", [{2}]"

Cause: The crs_stop command failed for the ASM instances on the node. This may be due to a CRS resource dependency or because the user has invalid credentials for stopping ASM instances.

Action: Determine whether there are database instances that depend on the ASM instance on the node by running the crs_stat -p ora.<db>.<inst>.inst command. If ASM instance resource names appear in the required resources for the database instance, then stop the database instances first using the 'srvctl stop instance' command. Do this before retrying the ASM instance stop operation or try to stop the ASM instance using SQL*Plus to obtain more diagnostic information.

PRKS-1013: Failed to stop ASM instances "{0}" on node "{1}", [{2}]"

Cause: See above.

Action: See above.

PRKS-1014: Failed to enable CRS resource for ASM instance "{0}" on node "{1}", [{2}]

Cause: Failed to set the enable flag of the ASM instance configuration in the OCR.

Action: Verify that the OCR is accessible by running the 'srvctl config ' command.

PRKS-1015: Failed to enable CRS resources for ASM instances "{0}" on node "{1}", [{2}]"

Cause: See above for each ASM instance.

Action: Check whether the OCR is accessible by running the 'srvctl config ' command.

PRKS-1016: Failed to disable CRS resource for ASM instance "{0}" on node "{1}", [{2}]"

Cause: Failed to reset the enable flag of the ASM instance configuration in the OCR.

Action: Verify that the OCR is accessible by running the 'srvctl config ' command.

PRKS-1017: Failed to disable CRS resources for ASM instances "{0}" on node "{1}", [{2}]"

Cause: See above for each ASM instance.

Action: Verify that the OCR is accessible by running the 'srvctl config ' command.

PRKS-1019: Cannot create CRS dependency between database instance "{0}" configured on node "{1}" and ASM instance "{2}" configured on node "{3}".

Cause: An attempt was made to create a CRS dependency between a database instance and an ASM instance that is configured on a different node.

Action: Make sure that the database instance and the ASM instance are on the same node before creating a CRS dependency between them.

PRKS-1020: Failed to create CRS dependency between database instance "{0}" and ASM instance "{1}", [{2}]

Cause: 'crs_register -u' failed to create CRS dependency between the database instance and the ASM instance on the node.

Action: Contact Oracle support.

PRKS-1021: Failed to remove CRS dependency between database instance "{0}" and ASM instance "{1}", [{2}]

Cause: 'crs_register -u' command failed to remove CRS dependency between the database instance and the ASM instance on the node.

Action: Contact Oracle Support.

PRKS-1022: Failed to remove CRS dependency between database instance "{0}" and ASM instances "{1}", [{2}]

Cause: See above.

Action: See above.

PRKS-1023: Failed to remove CRS resource for ASM instance "{0}" on node "{1}", [{2}]

Cause: crs_unregister command failed to unregister CRS resource for the ASM instances on the node.

Action: Contact Oracle Support.

PRKS-1026: ASM Configuration for node "{0}" does not exist in cluster registry."

Cause: An attempt was made to retrieve names of ASM instances configured on the node without configuring any ASM instance on it.

Action: First configure ASM instance using 'srvctl add asm' on the node command before executing the get configuration operation.

PRKS-1027: Configuration for ASM instance "{0}" does not exist in cluster registry."

Cause: An attempt was made to retrieve node name for a given ASM instance name which is not configured on any cluster nodes.

Action: First configure ASM instance using 'srvctl add asm' command before executing the get configuration operation.

PRKS-1028: Configuration for ASM instance "{0}" on node "{1}" does not exist in cluster registry.

Cause: An attempt was made to retrieve configuration for the ASM instance on the node name where it was not configured.

Action: Use 'srvctl config asm -n <node>' command to determine the ASM instance names configured on it and then pass one of these names as "-i <inst>" argument to 'srvctl config asm' command.

PRKS-1029: Client version "{0}" is not compatible with ASM instance configuration version "{1}" in cluster registry.

Cause: Version of the client that tried to retrieve ASM instance configuration is not compatible with ASM instance configuration version.

Action: Make sure client version is compatible with ASM instance configuration version before accessing it.

PRKS-1030: Failed to add configuration for ASM instance "{0}" on node "{1}" in cluster registry, [{2}]

Cause: Failed to add the configuration information for the ASM instance in the OCR.

Action: Verify that the OCR is accessible by executing 'srvctl config ' command.

PRKS-1031: "Failed to retrieve configuration for ASM instance "{0}" on node "{1}" from cluster registry, [{2}]

Cause: Failed to retrieve the configuration information for the ASM instance from the OCR.

Action: Verify that the OCR is accessible by executing 'srvctl config ' command.

PRKS-1032: Failed to modify configuration for ASM instance "{0}" on node "{1}" in cluster registry, [{2}]

Cause: Failed to modify the configuration information for the ASM instance on the node in the OCR.

Action: Verify that the OCR is accessible by executing 'srvctl config ' command.

PRKS-1033: Failed to remove configuration for ASM instance "{0}" on node "{1}" from cluster registry, [{2}]

Cause: Failed to remove the configuration information for the ASM instance from the OCR.

Action: Verify that the OCR is accessible by executing 'srvctl config ' command.

PRKS-1034: Failed to remove configuration for ASM instances "{0}" on node "{1}" from cluster registry, [{2}]"

Cause: See above.

Action: See above.

PRKS-1035: Failed to retrieve ORACLE_HOME value for ASM instance "{0}" on node "{1}" from cluster registry, [{2}]

Cause: Failed to retrieve the ORACLE_HOME from the ASM instance configuration in the OCR.

Action: Verify that the OCR is accessible by executing 'srvctl config ' command.

PRKS-1036: VIP resource for ASM instance node "{0}" does not exist."

Cause: The VIP resource for the node does not exist; it is a required resource for the ASM instance configuration on the node.

Action: Configure the VIP resource for the node using the 'srvctl add nodeapps' command as a privileged user before adding configuration for the ASM instance.

PRKS-1037: Failed to check existence of VIP resource for ASM instance node "{0}", [{1}]

Cause: 'crs_stat ora.<node>.vip' failed to check status of the VIP resource for the node.

Action: Contact Oracle Support.

PRKU—Command-Line Parser Utility Messages

PRKU-1000: "Exception Caused by: "

Cause: The current exception was caused by another earlier exception.

Action: Examine all of the nested exceptions to determine the root cause of the error.

PRKU-1001: "The parameter {0} is required."

Cause: The parameter specified is required by a command line utility.

Action: Supply the parameter specified.

PRKU-1002: "Unexpected argument {0}."

Cause: The argument specified is not needed or not understood by this utility.

Action: Verify the arguments on the command you are executing.

PRKU-1003: "The parameter {0} requires an argument."

Cause: The parameter specified must be accompanied by an argument.

Action: Verify the arguments on the command you are executing.

PRKV—Virtual IP Configuration Assistant Messages

PRKV-1013 "A network interface is required"

Cause: You did not provide a network interface for configuring the virtual IP address.

Action: When running VIPCA in GUI mode, select a suitable network interface from the list. When running VIPCA in silent mode, provide a suitable interface using the '-interfaces' option.

PRKV-1014 "Enter a valid IP address for the node \"{0}\"."

Cause: You did not enter an IP address to be configured as a virtual IP for the node.

Action: Enter an unused IP address to be used for configuring the virtual IP address.

PRKV-1015 "Enter a valid subnet mask for the IP address \"{0}\"."

Cause: You did not enter an IP netmask for the IP address.

Action: Enter a netmask for the IP address.

PRKV-1016 "IP address \"{0}\" is invalid. Enter a valid IP address."

Cause: IP address entered is invalid.

Action: Enter a valid IP address in standard IEEE format.

PRKV-1017 "IP address \"{0}\" does not exist. Enter a valid IP address."

Cause: IP address cannot be resolved in the network.

Action: Add an IP address in DNS or /etc/hosts so it can be resolved.

PRKV-1018 "IP address \"{0}\" has invalid format. Enter a valid IP address."

Cause: IP address is not in standard IEEE format.

Action: Enter a valid IP address in IEEE format.

PRKV-1019 "Netmask entered \"{0}\" has invalid format. Enter a valid netmask."

Cause: Netmask entered in not in standard IEEE format.

Action: Enter a valid IP netmask in IEEE format.

PRKV-1039 "IP address \"{0}\" has already been used. Enter an unused IP address."

Cause: The IP address entered is already used by another node as a virtual IP address.

Action: Enter an unused IP address.

PRKV-1059: "Invalid node name \"{0}\" entered in an input argument."

Cause: The IP address entered is already used by another node as a virtual IP address.

Action: Enter an unused IP address.

PRKV-1060: "Virtual IP address is not entered for node \"{0}\". Enter a valid virtual IP address."

Cause:

Action: Enter a valid virtual IP address for node {0} before re-attempting the command.

PRKV-1061 "Invalid IP address \"{0}\" entered in an input argument."

Cause:

Action: Enter valid IP address in place of {0} before reattempting command.

PRKV-1062 "Invalid netmask \"{0}\" entered in an invalid argument."

Cause:

Action: Enter valid netmask in place of {0} before reattempting command.

PRKV-1063 "Insufficient privileges."

Cause:

Action: Contact your system administrator to grant the appropriate privileges to you so that you can perform the required operations.

PRKV-1064 "Failed to obtain handle to CRS.

Cause:

Action: Refer to CRS logs for detailed error information.

PRKV-1065 "Host IP address \"{0}\" cannot be used as virtual IP for the node \"{1}\". Enter a different IP address.

Cause:

Action: Change IP address {0} before reattempting command.

PRKV-1066 "Host IP address \"{0}\" of cluster node \"{1}\" cannot be used as virtual IP for the node \"{2}\". Enter a different IP address."

Cause:

Action: Change IP address {0} before reattempting command.

PRKV-1070: Different interface names entered as an input for each node. Specify same set of interfaces for each node in the cluster.

Cause: The nodes in a cluster should use the same set of public interfaces for configuring a Virtual IP.

Action: Make sure that you have public interfaces on each node of the cluster and that their names and subnets match.

PRKV-1072: Invalid syntax used for option \"nodevips\". Check usage (vipca -help) for proper syntax.

Cause: The syntax argument value specified for option "nodevips" is invalid.

Action: Run "vipca -help" to see the syntax in which nodevips option value should be specified.

PRKV-1073: "Node names \"{0}\" are duplicated in an input argument."

Cause: The node names entered as input to the VIPCA command line should be unique.

Action: Remove duplicate node names from VIPCA command line and rerun the command.

PRKV-1073: "Node names \"{0}\" are duplicated in an input argument."

Cause: The node names that you enter as input to the VIPCA command line should be unique.

Action: Remove duplicate node names from the VIPCA command line and rerun the command.

PRKV-1074: The given interface(s), \"{0}\" is not public. Public interfaces should be used to configure virtual Ips.

Cause: The interface that you have specified as input is not a public interface. The Oracle Clusterware uses public interfaces for configuring VIPs.

Action: Find a public interface on your system with a name and subnet that matches all of the nodes of the cluster and specify this interface as an argument to VIPCA.

Glossary

Automatic Workload Repository (AWR)

A built-in repository that exists in every Oracle Database. At regular intervals, the Oracle Database makes a snapshot of all of its vital statistics and workload information and stores them in the AWR.

cache coherency

The synchronization of data in multiple caches so that reading a memory location through any cache will return the most recent data written to that location through any other cache. Sometimes called cache consistency.

Cache Fusion

A diskless cache coherency mechanism in Oracle Real Application Clusters that provides copies of blocks directly from a holding instance's memory cache to a requesting instance's memory cache.

cluster

Multiple interconnected computers or servers that appear as if they are one server to end users and applications.

cluster file system

A distributed file system that is a cluster of servers that collaborate to provide high performance service to their clients. Cluster file system software deals with distributing requests to storage cluster components.

cluster database

The generic term for a Oracle Real Application Clusters database.

CRSD - Oracle Clusterware Daemon

The primary Oracle Clusterware process.

Cluster Synchronization Services (CSS)

An Oracle Clusterware component that discovers and tracks the membership state of each node by providing a common view of membership across the cluster. CSS also monitors process health, specifically the health of the database instance. The Global Enqueue Service Monitor (LMON), a background process that monitors the health of the cluster database environment and registers and de-registers from CSS. See also, OCSSD.

Cluster Verification Utility (CVU)

A tool that verifies a wide range of Oracle RAC components such as shared storage devices, networking configurations, system requirements, Oracle Clusterware, groups, and users.

CRSD

A Linux or UNIX process that performs high availability recovery and management operations such as maintaining the OCR. Also manages application resources and runs as `root` user (or by a user in the `admin` group on Mac OS X-based systems) and restarts automatically upon failure.

Distributed Transaction Processing (DTP)

The paradigm of distributed transactions, including both XA-type externally coordinated transactions, and distributed-SQL-type (database links in Oracle) internally coordinated transactions.

Enterprise Manager Configuration Assistant (EMCA)

A graphical user interface-based configuration assistant that you can use to configure Enterprise Manager features.

Event Manager (EVM)

The background process that publishes Oracle Clusterware events. EVM scans the designated callout directory and runs all scripts in that directory when an event occurs.

Event Manager Daemon (EVMD)

A Linux or UNIX event manager daemon that starts the `racgevt` process to manage callouts.

extended distance cluster

A cluster where the nodes in the cluster are separated by greater distances from two buildings across the street, to across a campus or across a city. For availability reasons, the data needs to be located at both sites, and therefore one needs to look at alternatives for mirroring the storage.

Fast Application Notification (FAN)

Applications can use FAN to enable rapid failure detection, balancing of connection pools after failures, and re-balancing of connection pools when failed components are repaired. The FAN notification process uses system events that Oracle publishes when cluster servers become unreachable or if network interfaces fail.

Fast Connection Failover

Fast Connection Failover provides high availability to FAN integrated clients, such as clients that use JDBC, OCI, or ODP.NET. If you configure the client to use fast connection failover, then the client automatically subscribes to FAN events and can react to database `UP` and `DOWN` events. In response, Oracle gives the client a connection to an active instance that provides the requested database service.

General Parallel File System (GPFS)

General Parallel File System (GPFS) is a shared-disk IBM file system product that provides data access from all of the nodes in a homogenous or heterogeneous cluster.

forced disk write

In Real Application Clusters, a particular data block can only be modified by one instance at a time. If one instance modifies a data block that another instance needs, then whether a forced disk write is required depends on the type of request submitted for the block.

Global Cache Service (GCS)

Process that implement Cache Fusion. It maintains the block mode for blocks in the global role. It is responsible for block transfers between instances. The Global Cache Service employs various background processes such as the Global Cache Service Processes (LMSn) and Global Enqueue Service Daemon (LMD).

Global Cache Service Processes (LMSn)

Processes that manage remote messages. Oracle RAC provides for up to 10 Global Cache Service Processes.

Global Cache Service (GCS) resources

Global resources that coordinate access to data blocks in the buffer caches of multiple Oracle RAC instances to provide cache coherency.

global database name

The full name of the database that uniquely identifies it from any other database. The global database name is of the form database_name.database_domain—for example: OP.US.ORACLE.COM

global dynamic performance views (GV$)

Dynamic performance views storing information about all open instances in a Real Application Clusters cluster. (Not only the local instance.) In contrast, standard dynamic performance views (`V$`) only store information about the local instance.

Global Enqueue Service (GES)

A service that coordinates enqueues that are shared globally.

Global Enqueue Service Daemon (LMD)

The resource agent process that manages requests for resources to control access to blocks. The LMD process also handles deadlock detection and remote resource requests. Remote resource requests are requests originating from another instance.

Global Enqueue Service Monitor (LMON)

The background LMON process monitors the entire cluster to manage global resources. LMON manages instance deaths and the associated recovery for any failed instance. In particular, LMON handles the part of recovery associated with global resources. LMON-provided services are also known as Cluster Group Services.

Global Services Daemon (GSD)

A component that receives requests from `SRVCTL` to execute administrative job tasks, such as startup or shutdown. The command is executed locally on each node, and the results are returned to `SRVCTL` by default.

High Availability Cluster Multi-Processing (HACMP)

High Availability Cluster Multi-Processing is an IBM AIX-based high availability cluster software product. HACMP has two major components: high availability (HA) and cluster multi-processing (CMP).

Oracle Hardware Assisted Resilient Data (HARD)

The Oracle Hardware Assisted Resilient Data (HARD) Initiative prevents data corruptions. The HARD initiative uses Oracle data validation algorithms inside storage devices to prevent writing corrupted data to permanent storage.

high availability

Systems with redundant components that provide consistent and uninterrupted service, even in the event of hardware or software failures. This involves some degree of redundancy.

instance

For an Oracle RAC database, each node in a cluster usually has one instance of the running Oracle software that references the database. When a database is started, Oracle allocates a memory area called the System Global Area (SGA) and starts one or more Oracle processes. This combination of the SGA and the Oracle processes is called an instance. Each instance has unique Oracle System Identifier (sid), instance name, rollback segments, and thread ID.

instance membership recovery

The method used by Oracle RAC guaranteeing that all cluster members are functional or active. instance membership recovery polls and arbitrates the membership. Any members that do not show a heartbeat by way of the control file or who do not respond to periodic activity inquiry messages are presumed terminated.

instance name

Represents the name of the instance and is used to uniquely identify a specific instance when clusters share common services names. The instance name is identified by the `INSTANCE_NAME` parameter in the instance initialization file, `initsid.ora`. The instance name is the same as the Oracle System Identifier (sid).

instance number

A number that associates extents of data blocks with particular instances. The instance number enables you to start up an instance and ensure that it uses the extents allocated to it for inserts and updates. This will ensure that it does not use space allocated for other instances.

interconnect

The communication link between nodes.

Logical Volume Manager (LVM)

A generic term that describes Linux or UNIX subsystems for online disk storage management.

Interprocess Communication (IPC)

A high-speed operating system-dependent transport component. The IPC transfers messages between instances on different nodes. Also referred to as the interconnect.

Master Boot Record (MBR)

A program that executes when a computer starts. Typically, the MBR resides on the first sector of a local hard disk. The program begins the startup process by examining the partition table to determine which partition to use for starting the system. The MBR program then transfers control to the boot sector of the startup partition, which continues the startup process.

Network Attached Storage (NAS)

Storage that is attached to a server by way of a network.

Network Time Protocol (NTP)

An Internet standard protocol, built on top of TCP/IP, that ensures the accurate synchronization to the millisecond of the computer clock times in a network of computers.

Network Interface Card (NIC)

A card that you insert into a computer to connect the computer to a network.

node

A node is a computer system on which an instance resides.

Oracle Cluster File System (OCFS)

The Oracle proprietary cluster file system software that is available for Linux and Windows platforms.

Oracle Cluster Registry (OCR)

The Oracle RAC configuration information repository that manages information about the cluster node list and instance-to-node mapping information. The OCR also manages information about Oracle Clusterware resource profiles for customized applications.

Object Link Manager (OLM)

The Oracle interface that maps symbolic links to logical drives and displays them in the OLM graphical user interface.

OCLSMON

A process that runs on nodes in Oracle RAC environments.`OCLSOMON` monitors CSS hangs that result from load or scheduling issues. If `OCLSOMON` detects either of these, the process shuts down the affected node to prevent database corruption.

OCLSVMON

OCLSVMON is a process that runs when vendor clusterware is installed. Basically due to the existence of this process, the Oracle Clusterware can delay log flushes. This delay is useful in diagnosing problems because the vendor clusterware allows a delay of eviction processing until after log files have flushed.

OCSSD

A Linux or UNIX process that manages the Cluster Synchronization Services (CSS) daemon. Manages cluster node membership and runs as `oracle` user; failure of this process results in cluster restart.

Oracle Interface Configuration Tool (OIFCFG)

A command-line tool for both single-instance Oracle databases and Oracle RAC databases that enables you to allocate and de-allocate network interfaces to components, direct components to use specific network interfaces, and retrieve component configuration information. The Oracle Universal Installer (OUI) also uses OIFCFG to identify and display available interfaces.

Oracle Notification Services (ONS)

A publish and subscribe service for communicating information about all FAN events.

OPROCD

A Linux or UNIX process monitor for a cluster. Note that this process will only appear on platforms that do not use vendor clusterware with Oracle Clusterware.

Oracle Clusterware

This is clusterware that is provided by Oracle to manage cluster database processing including node membership, group services, global resource management, and high availability functions.

Oracle Universal Installer (OUI)

A tool to install Oracle Clusterware, the Oracle relational database software, and the Oracle Real Application Clusters software. You can also use the Oracle Universal Installer to launch the Database Configuration Assistant (DBCA).

raw device

A disk drive that does not yet have a file system set up. Raw devices are used for Oracle Real Application Clusters because they enable the sharing of disks. See also raw partition.

raw partition

A portion of a physical disk that is accessed at the lowest possible level. A raw partition is created when an extended partition is created and logical partitions are assigned to it without any formatting. Once formatting is complete, it is called a cooked partition. See also raw device.

Recovery Manager (RMAN)

An Oracle tool that enables you to back up, copy, restore, and recover datafiles, control files, and archived redo logs. It is included with the Oracle server and does not require separate installation. You can invoke RMAN as a command line utility from the operating system (O/S) prompt or use the GUI-based Enterprise Manager Backup Manager.

Runtime Connection Load Balancing

Enables Oracle to make intelligent service connection decisions based on the connection pool that provides the optimal service for the requested application based on current workloads. The JDBC, ODP.NET, and OCI clients are integrated with the load balancing advisory; you can use any of these client environments to provide runtime connection load balancing.

scalability

The ability to add additional nodes to Real Application Clusters applications and achieve markedly improved scale-up and speed-up.

Secure Shell (SSH)

A program for logging into a remote computer over a network. You can use SSH to execute commands on a remote system and to move files from one system to another. SSH uses strong authentication and secure communications over insecure channels.

Server Control (SRVCTL) Utility

Server Management (SRVM) comprises the components required to operate Oracle Enterprise Manager in Oracle Real Application Clusters. The SRVM components, such as the Intelligent Agent, Global Services Daemon, and `SRVCTL`, enable you to manage

cluster databases running in heterogeneous environments through an open client/server architecture using Oracle Enterprise Manager.

services

Entities that you can define in Oracle RAC databases that enable you to group database workloads and route work to the optimal instances that are assigned to offer the service.

shared everything

A database architecture in which all instances share access to all of the data.

singleton services

Services that run on only one instance at any one time. By defining the Distributed Transaction Property (DTP) property of a service, you can force the service to be a singleton service.

split brain syndrome

Where two or more instances attempt to control a cluster database. In a two-node environment, for example, one instance attempts to manage updates simultaneously while the other instance attempts to manage updates.

system identifier (SID)

The Oracle system identifier (SID) identifies a specific instance of the running Oracle software. For a Real Application Clusters database, each node within the cluster has an instance referencing the database.

transparent application failover (TAF)

A runtime failover for high-availability environments, such as Real Application Clusters and Oracle Real Application Clusters Guard, TAF refers to the failover and re-establishment of application-to-service connections. It enables client applications to automatically reconnect to the database if the connection fails, and optionally resume a SELECT statement that was in progress. This reconnect happens automatically from within the Oracle Call Interface (OCI) library.

voting disk

A file that manages information about node membership.

Index

Symbols

$ORACLE_HOME/root.sh script, 7-8

A

ACMS
  Atomic Controlfile to Memory Service, 1-6
Active Session History, Oracle RAC, 11-8
Active Session History, Top Cluster Events, 11-8
Active Session History, Top Remote Instance, 11-8
ACTIVE_INSTANCE_COUNT initialization parameter, 3-13
add node
  ASM instances for shared storage, 9-5
  using NETCA to add a listener, 9-8
adding a node
  root.sh script, 9-8
ADDM
  global monitoring, 11-7
  see Automatic Database Diagnostic Monitor
ADDM for Oracle Real Application Clusters mode, 11-7
administering
  services, 4-24
  services with SRVCTL, 4-29
administering instances
  with Server Management, 3-1
administering services
  DBCA, 4-26
  Enterprise Manager, 4-26
  PL/SQL, 4-26
  SRVCTL, 4-26
administrative tools
  overview and concepts, 1-11
  SYSASM privilege, 3-6
ADRCI
  ADR Command-Line Interpreter, 0-xv, B-2
Advanced Queuing
  and FAN, 4-8
affinity
  awareness, 6-4
aggregates
  by instances, 11-4
  by services, 11-4
  by waits, 11-4
alert administration
  Enterprise Manager, 3-21
alert blackouts
  Enterprise Manager, 3-22
alert logs, B-2
  managing, B-1
ALTER SYSTEM ARCHIVE LOG CURRENT statement, 3-3
ALTER SYSTEM ARCHIVE LOG statement, 3-3
  INSTANCE option, 3-3
ALTER SYSTEM CHECKPOINT LOCAL statement, 3-3
ALTER SYSTEM CHECKPOINT statement
  global versus local, 3-3
  specifying an instance, 3-3
ALTER SYSTEM KILL SESSION statement
  terminating a session on a specific instance, 3-7
ALTER SYSTEM QUIESCE RESTRICTED statement
  quiescing a single-instance database, 3-15
ALTER SYSTEM statement
  CHECKPOINT clause, 3-3
ALTER SYSTEM SWITCH LOGFILE statement, 3-3
applications
  consolidating multiple in a single database, 10-3
  highly available, 10-1
  scalability, 10-4
  spanning XA transactions across Oracle RAC instances, 4-22
  using pre-created database sessions, 4-16
AQ_HA_NOTIFICATIONS parameter
  enabling event notification, 4-16
ARCHIVE LOG command
  SQL*Plus, 3-3
ARCHIVE_LAG_TARGET initialization parameter, 3-13, 3-14
archived redo log files
  file format and destination, 5-5
  file naming, 5-4
  log sequence number, 5-5
archiver process
  monitor, 5-9
archiving mode
  changing, 5-9
ASH reports, 11-8
ASM
  see Automatic Storage Management (ASM)

asm
SRVCTL object noun name, A-5
ASM instances
administering with SRVCTL, 2-9
ASM_PREFERRED_READ_FAILURE_GROUPS initialization parameter, xv, 2-5, 3-12
Atomic Controlfile to Memory Service (ACMS), 1-6
Automatic Database Diagnostic Monitor, 11-9, 11-10, 11-11
Automatic Database Diagnostic Monitor (ADDM), 1-13
analyzing AWR data, 11-7
Automatic Database Diagnostic Monitoring (ADDM), xiv, 11-7
Automatic Diagnostic Repository (ADR)
ADRCI command-line interpreter, 0-xv, B-2
automatic load balancing
configuring RMAN channels for multiple instances, 5-3
automatic segment space management (ASSM), 10-5
automatic segment-space management
tablespace use in Oracle RAC, 10-5
Automatic Storage Management (ASM), 2-4
archiving scenario, 5-6
ASM preferred read failure groups, xv, 2-5
cloning, 7-1
converting single-instance ASM to clustered ASM, 2-6
converting single-instance to a cluster storage manager, 2-6
creating instances with DBCA, 7-6
installation, 1-9
instance creation on newly added nodes, 7-1, 9-5
preferred mirror read disks, 2-5
preferred read, xv
preferred read disks, 3-12
rolling upgrade, xv, 2-5
rolling upgrades, xv
storage solution, 1-4, 2-1
SYSASM privilege, 3-6
automatic undo management
tablespace use in Oracle RAC, 10-5
automatic workload management
concepts, 1-7, 4-3
description, 4-1
manual rebalancing, 4-2
Automatic Workload Repository, 4-2, 4-3, 4-30, 11-7, 11-10
snapshots, 11-7
Automatic Workload Repository (AWR)
monitoring performance, 4-4
AVAILABLE instances
for services, 4-3
Average Active Sessions chart
performance monitoring, 11-4
AWR
see Automatic Workload Repository

B

background processes
Oracle RAC, 1-6
SMON, 3-5, 6-3
background thread trace files, B-2
backups
server parameter file, 3-11
bandwidth
interconnect, 11-5
benefits
of cloning Oracle Clusterware, 7-2
best practices
deploying Oracle RAC for high availability, 10-2
block mode conversions
statistics for, 11-6
blocks
associated with instance, 6-3
buffer cache, 1-6
instance recovery, 6-3
buffer sizes
IPC, adjusting for Oracle Real Application Clusters, 11-5

C

Cache Fusion, 1-6
and e-commerce applications, 10-6
overview, 1-13
callouts
how they are run, 4-8
capacity
increasing, 10-4
catclustdb.sql script, 1-13
CATCLUST.SQL script
using to create views for Oracle Real Application Clusters, 11-6
channels
configure one RMAN channel for each Oracle RAC instance, 5-3
configuring during crosscheck or restore operations, 5-3
configuring for RMAN, 5-3
chart
Global Cache Block Access Latency, 11-3
charts
Average Active Sessions, 11-4
Cluster Host Load Average, 11-3
Database Throughput, 11-4
CLB_GOAL_LONG, 4-5
CLB_GOAL_SHORT, 4-5, 4-8
client
application environments and FAN, 4-12
clients
integrated for FAN events, 4-7
JDBC thick, 4-13
JDBC thin, 4-13
client-side
load balancing, 4-5
client-side load balancing, 4-6
clone.pl script

cloning parameters, 7-5, 7-8
environment variables, 7-5, 7-8
cloning, 1-10, 7-1, 7-3
benefits, 7-2
deployment phase, 7-3
LD_LIBRARY_PATH environment variable, 7-5, 7-8
log files, 7-9
parameters passed to the clone.pl script, 7-5, 7-8
preparation phase, 7-2
running $ORACLE_HOME/root.sh script, 7-8
setting ORACLE_BASE, 7-5, 7-8
setting ORACLE_HOME, 7-5, 7-8
setting ORACLE_HOME_NAME, 7-5, 7-8
cluster
definition of, 1-1
Cluster Database Home Page, 11-2
Cluster Database Performance page
Top Activity drill down menu, 11-4
cluster file system
archiving parameter settings, 5-6
archiving scenario, 5-6
restore, 6-2
storage in Real Application Clusters, 2-1
Cluster Host Load Average page
cluster database performance, 11-3
Cluster Managed Database Services Detail Page
Enterprise Manager, 4-27
Cluster Managed Database Services Page
Enterprise Manager, 4-27
CLUSTER NODE NAME parameter
FAN, and matching database signature, 4-10
cluster nodes name
in clone.pl script, 7-5, 7-8
Cluster Ready Services (CRS)
described, 1-3
CLUSTER_DATABASE initialization parameter, 3-12, 3-13
CLUSTER_DATABASE_INSTANCE initialization parameter, 3-13
CLUSTER_DATABASE_INSTANCES initialization parameter, 3-12
CLUSTER_INTERCONNECTS
parameter, 11-5
CLUSTER_INTERCONNECTS parameter
examples, 3-17
recommendations for using, 3-16
CLUSTER_NODES parameter
in clone.pl script, 7-5, 7-8
clustered ASM
converting a single-instance ASM, 2-6
clusters
consolidating multiple databases in, 10-3
clusterware management solution, 1-3
CLUSTERWIDE_DISTRIBUTED_TRANSACTIONS initialization parameter
spanning XA transactions across Oracle RAC instances, xiv
CMAN session pools
and FAN, 4-12
command-line interpreter
ADR Command-Line Interpreter (ADRCI), 0-xv, B-2
committed data
instance failure, 6-3
communication protocols
verifying settings for, 11-5
compatibility
Oracle RAC and Oracle Database software, 1-9
COMPATIBLE initialization parameter, 3-13
configuring channels
during restore or crosscheck operations, 5-3
configuring preferred mirror read disks in extended clusters, 2-5
CONNECT command, 3-2, 3-3
SQL*Plus, 3-3
CONNECT SYS
example of, 3-6
connecting
to instances, 1-12, 3-2
connection failover callback, 4-18
connection load balancing
concepts, 1-8
introduction to, 4-1
long method, 4-5
short method, 4-5
connection pools
and FAN, 4-13
consistent blocks, 1-6
CONTROL_FILES initialization parameter, 3-13
CREATE PFILE
FROM MEMORY clause, 3-9
Create Services Page
Enterprise Manager, 4-27
CREATE SPFILE
FROM MEMORY clause, 3-9
CREATE_SERVICE procedure
DBMS_SERVICE PL/SQL package, 4-28
creating
server parameter files, 3-9, 3-11
services, 4-24
crosscheck operations
configuring channels during, 5-3
crosschecking on multiple nodes
RMAN backups, 5-3
CRS resources
management of, 1-3
current blocks, 1-6
CVU
overview and concepts, 1-12

D

data dictionary
querying views, 11-6
data security
wallet, 10-8
data warehouse
deploying applications for in Oracle Real Application Clusters, 10-7

database
creation, 1-9
quiescing, 3-15
SRVCTL object noun name, A-5
Database Configuration Assistant (DBCA), 1-9
adding and deleting instances in silent mode, 9-10
cloning Oracle RAC instances, 7-8
creating views for Oracle Real Application Clusters, 11-6
Database Storage page, 9-10
Instance Management page, 9-9
List of Cluster Databases page, 9-9
Operations page, 9-9
running the catclustdb.sql script, 1-13
Summary dialog, 9-10
Welcome page, 9-9
database signatures
matching the FAN parameters, 4-9
Database Storage page, 9-10
Database Throughput page
performance monitoring, 11-4
DATABASE UNIQUE NAME parameter
FAN, and matching database signature, 4-9
databases
consolidating multiple in a cluster, 10-3
controlling restarts, 3-18
scalability, 10-4
databases sessions
pre-created, 4-16
Datagram Protocol (UDP), 1-5, B-3
DB_BLOCK_SIZE initialization parameter, 3-13
DB_DOMAIN initialization parameter, 3-13
DB_FILES initialization parameter, 3-14
DB_NAME initialization parameter, 3-12, 3-14
DB_RECOVERY_FILE_DEST initialization parameter, 3-14
DB_RECOVERY_FILE_DEST_SIZE initialization parameter, 3-14
DB_UNIQUE_NAME initialization parameter, 3-14
DBCA
creating ASM instances, 7-6
see Database Configuration Assistant (DBCA)
DBMS_SERVICE, 4-2
DBMS_SERVICE package
procedures for administering services, 4-28
DBMS_SERVICE.GOAL_SERVICE_TIME procedure
service goal, 4-16
DBMS_SERVICE.MODIFY_SERVICE procedure
enabling event notification, 4-16
default database service, 1-10, 3-13, 4-26
degree of parallelism (DOP), 10-7
DELETE_SERVICE procedure
DBMS_SERVICE PL/SQL package, 4-29
dependencies
and services, 4-4
deploying
Oracle RAC database software, 7-6
Oracle Real Application Clusters environments, 1-11, 10-1
design
Oracle Real Application Clusters environments, 1-11, 10-1
diagnosing problems using ADR, B-2
DIAGNOSTIC_DEST parameter, B-2
DISCONNECT command, 3-2
DISCONNECT_SESSION procedure
DBMS_SERVICE PL/SQL package, 4-29
disconnecting from instances, 3-2
DISPATCHERS initialization parameter, 3-12
specifying a service with, 4-26
distributed SQL transactions, 4-23
Distributed Transaction Processing, 4-22, 10-6
distributed transactions, 4-22, 10-6
directing to a single instance in the cluster, 4-22
services in Oracle RAC, 4-22
XA transactions span instances, xiv, 4-22
DML_LOCKS initialization parameter, 3-14
DTP, 4-22, 10-6
DTP Service., 4-22
DTP/XA transactions, 4-23
dynamic performance views
creating, 11-6
for performance monitoring, 11-6
dynamic resource allocation
overview, 1-13

E

e-commerce
applications in Oracle Real Application Clusters, 10-6
ENCRYPTION_WALLET_LOCATION parameter, 10-8
Enterprise Manager, 3-2
administering services, 4-26
alert administration, 3-21
alert blackouts, 3-22
Cluster Database page, 0-xvii, 11-2
job administration, 3-21
overview, 1-3
overview and concepts, 1-11
services pages, accessing, 4-27
Enterprise Manager Grid Control
instance discovery, 3-20
node discovery, 3-20
environment variables
passed to the clone.pl script, 7-5, 7-8
error messages
for management tools, C-1
PRKA, C-2
PRKC, C-4
PRKD, C-14
PRKE, C-14
PRKH, C-15
PRKI, C-16
PRKN, C-18
PRKO, C-18
PRKP, C-22
PRKR, C-29

PRKS, C-35
PRKU, C-39
PRKV, C-39
event notification
enabling, 4-16
examples
setting the CLUSTER_INTERCONNECTS parameter, 3-17
extended distance cluster
ASM preferred mirror read, xv, 2-5
extended distance clusters
configuring preferred mirror read disks, 2-5
external transaction managers
OraMTS, 4-23

F

failover callback routine, 4-21
failure
instance, 6-2
multiple node, 6-3
node, 6-3
failure events
calling routines with SQL_ORCLATTR_FAILOVER_CALLBACK, 4-18
identifying with the fo_code parameter, 4-19
FAN events
enabling for JDBC, 4-13
Fast Application Notification (FAN), 4-7
and high availability events, 4-9
callouts definition, 4-10
callouts, how to use, 4-10
events, enabling for JDBC clients, 4-13
events, enabling for OCI, 4-15
events, enabling for ODP.NET, 4-16
events, enabling ODP.NET clients, 4-17
how events are published, 4-8
introduction, 4-2
overview, 4-7
parameters and matching database signatures, 4-9
uses, 4-7
Fast Connection Failover
enabling JDBC clients, 4-13
enabling with thin and thick clients, 4-13
introduction to, 4-2
fault diagnosability, B-2
features, new, xiii
files
archived redo log files, 5-5
redo log, 5-5
fo_code parameter
identifying failure events, 4-19
FROM MEMORY clause
on CREATE PFILE or CREATE SPFILE, 3-9

G

GC_SERVER_PROCESSES initialization parameter
specifying the number of LMSn processes, 10-3
GCS_SERVER_PROCESSES initialization parameter, 3-12
Global ADDM mode, 11-7
Global Cache Block Access Latency
performance monitoring, 11-3
Global Cache Service (GCS), 1-6
Global Cache Service Process (LMS), 1-6
Global Cache Service Processes (LMSn)
reducing the number of, 10-3
specifying the number of, 10-3
Global Cache Service statistics, 11-9, 11-10
GLOBAL clause
forcing a checkpoint, 3-3
Global Enqueue Service (GES), 1-6
Global Enqueue Service Daemon (LMD), 1-6
Global Enqueue Service Monitor (LMON), 1-6
Global Enqueue Service statistics, 11-9
global performance data
with ADDM, xiv, 11-7
Global Resource Directory (GRD), 1-6
Global Services Daemon (GSD), 1-3
Global Transaction Process (GTX0-j), 1-6
GLOBAL_TXN_PROCESSES initialization parameter, 4-22
goals
and the load balancing advisory, 4-11
for load balancing advisory, 4-11
GTX0-j
Global Transaction Process, 1-6
GV$SESSION dynamic performance view, 3-8

H

hardware-security module (HSM) devices
used by wallets, 10-8
hash partitioning
with Oracle Real Application Clusters, 10-5
high availability
best practices, 10-2
for Oracle RAC Databases, 10-1
high availability framework
concepts, 1-7
introduction to, 4-2
HOST command, 3-4
SQL*Plus, 3-4

I

initdb_name.ora file
DIAGNOSTIC_DEST parameter, B-2
initialization parameters
cluster database issues regarding, 3-11
CLUSTER_INTERCONNECTS, 11-5
RECOVERY_PARALLELISM, 6-6
settings for instances, 3-9
that must be identical on all instances, 3-13
that must be unique on all instances, 3-14
INST_ID column, 11-6
installation
introduction, 1-8

Oracle Real Application Clusters, 1-9
installations
performing multiple simultaneous cluster, 7-1
instance
SRVCTL object noun name, A-5
instance discovery
Enterprise Manager Grid Control, 3-20
Instance Enqueue Process (LCK0), 1-7
Instance Management page, 9-9
INSTANCE NAME initialization parameter, 3-12
INSTANCE option, 3-3
INSTANCE parameter
FAN, and matching database signature, 4-9
INSTANCE_NAME initialization parameter, 3-14
INSTANCE_NUMBER initialization parameter, 3-14
INSTANCE_TYPE initialization parameter, 3-14
instances
aggregated for service performance, 11-4
cloning Oracle RAC, 7-8
effect of SQL*Plus commands on, 3-3
failure, 6-3
failures, recovery from, 6-2
initialization parameter settings, 3-9
maximum number for Oracle RAC, 1-1
memory structures, 1-5
parallel processes, 4-25
private interconnect usage, B-3
recovery, 3-5, 6-3
recovery, abnormal shutdown, 3-5
recovery, multiple failures, 6-3
Server Management, 3-1
shutting down, 3-5
terminating session on, 3-7
verifying, 3-7
instead of Statspack, 1-13
interconnect
and performance, 11-5
and the Oracle RAC architecture, 1-1, 1-4
bandwidth, 11-5
protocols for Oracle Real Application Clusters, 11-5
verifying settings for, 11-5
interconnects
alternatives to the private network, 3-16
private, B-3
Interconnects page
monitoring clusterware with Oracle Enterprise Manager, 11-2
monitoring Oracle Clusterware, 11-3
IPCs
buffer sizes, adjusting, 11-5

J

Java
enabling tools for tracing, B-3
Java Database Connectivity (JDBC) clients
ONS usage, 4-4
JDBC
and FAN, 4-12
enabling FAN events for, 4-13
thick clients, 4-13
thin clients, 4-13
JDBC clients
enabling for Fast Connection Failover, 4-13
job administration
Enterprise Manager, 3-21
job scheduling
with Oracle Scheduler, xvii

K

KILL SESSION clause
on the ALTER SYSTEM statement, 3-7

L

LCK0 Instance Enqueue Process, 1-7
LD_LIBRARY_PATH environment variable
setting during cloning, 7-5, 7-8
LICENSE_MAX_USERS initialization parameter, 3-15
List of Cluster Databases page, 9-9
listener object
SRVCTL object noun name, A-5
listeners
adding during add node, 9-8
creating during cloning, 7-6
Oracle Net, 1-3
removing with srvctl remove listener command, A-36
LMD Global Enqueue Service Daemon, 1-6
LMON Global Enqueue Service Monitor, 1-6
LMS Global Cache Service Process, 1-6
LMS processes
reducing the number of, 10-3
LMSn processes
reducing the number of, 10-3
load balancing
by services, 1-4
OCI runtime connection, 4-16
OCI runtime connection load balancing, xiv
server-side, 4-5
Load Balancing Advisory, 4-24
load balancing advisory
and FAN events, 4-12
concepts, 1-7
configuring your environment for using, 4-11
deployment, 4-11
description of, 4-10
events and FAN, 4-7
introduction to, 4-2
Local ADDM mode, 11-7
local archiving scenario
RMAN, 5-7
Local Area Network (LAN), 1-4
LOCAL clause
forcing a checkpoint, 3-3
local file system
archiving parameter settings, 5-7

restore, 6-2
local node name
in clone.pl script, 7-5, 7-8
LOCAL_NODE parameter
in clone.pl script, 7-5, 7-8
location transparency
using service names, 4-26
log files
tracing, B-3
log sequence numbers, 5-5
LOG_ARCHIVE_FORMAT initialization parameter, 3-15
LOG_ARCHIVE_FORMAT parameter, 5-5

M

mass deployment
cloning, 1-10, 7-1, 7-3
media failures
recovery from, 6-5
memory structures
in Oracle RAC, 1-5
message request counters, 11-6
messages
errors for management tools, C-1
migration
application, 10-4
mission critical systems
considerations for Oracle RAC, 10-1
modified data
instance recovery, 6-3
MODIFY_SERVICE procedure
DBMS_SERVICE PL/SQL package, 4-29
monitoring
archiver process, 5-9
overview and concepts, 1-12
performance of global cache block access, 11-3
statistics for Oracle Real Application Clusters, 11-6
monitoring host load average, 11-3
multiple node failures, 6-3
multiple nodes
starting from one node, 3-6
multiplexed redo log files, 2-2

N

Net Configuration Assistant
see NETCA
Net Configuration Assistant (NETCA)
adding a listener, 9-8
NETCA, 1-9
creating listeners during cloning, 7-6
Network Attached Storage (NAS), 1-5
network file system, 1-4
new features, xiii
NIC bonding, 3-16
node
failure and VIP addresses, 1-5
node discovery
Enterprise Manager Grid Control, 3-20
nodeapps
SRVCTL object noun name, A-5
nodes
affinity awareness, 6-4
failure of, 6-3
virtual IP addresses, A-3
nosharedhome argument, 9-14

O

objects
creation of and effect on performance, 10-6
srvctl nouns and abbreviations, A-5
OCI environment
to receive service metrics, 4-16
OCI session pooling, 4-16
OCI Session Pools
and FAN, 4-12
OCI session pools
optimizing, 4-16
runtime connection load balancing, 4-16
service metrics, 4-16
OCI_EVENTS mode
setup for runtime connection load balancing, 4-16
OCI_SPC_NO_RLB mode parameter
setting for runtime connection load balancing, 4-16
OCI_THREADED mode
setup for runtime connection load balancing, 4-16
OCISessionPoolCreate()
setting OCI_SPC_NO_RLB mode parameter, 4-16
ocr.loc file
on a shared file system, 9-15
ODBC applications
calling SQL_ORCLATTR_FAILOVER_HANDLE attribute, 4-19
ODBC_FO_ABORT failure event, 4-19
ODBC_FO_BEGIN failure event, 4-19
ODBC_FO_END failure event, 4-19
ODBC_FO_ERROR failure event, 4-19
ODBC_FO_REAUTH failure event, 4-19
ODP.NET
and FAN, 4-12
and FAN events, 4-16
and Fast Connection Failover, 4-17
load balancing advisory events, 4-17
OIFCFG command-line interface
run during add node, 9-8
online recovery, 6-3
online transaction processing
applications in Oracle Real Application Clusters, 10-6
Operations page, 9-9
Oracle Call Interface (OCI)
runtime connection load balancing, xiv, 4-16
Oracle Cluster File System (OCFS), 1-4
configuring when extending Oracle RAC, 9-5
Oracle Clusterware
cloning, 1-10

controlling database restarts, 3-18
described, 1-3
introduction, 1-3
introduction and concepts of, 1-1
managing Oracle processes, 1-3
Oracle Enterprise Manager
adding database instances to nodes, 9-9
Automatic Database Diagnostic Monitoring (ADDM), xiv, 11-7
Cluster Database Performance page
Cluster Database Performance page
performance statistics for an Oracle RAC database, 0-xviii, 11-3
deleting database instances from nodes, 9-11
Interconnects page, 11-3
using the Interconnects page to monitor Oracle Clusterware, 11-2
Oracle home
adding in silent mode, 9-8
cloning, 1-10
Oracle Maximum Availability Architecture (MAA), 10-2
Oracle Net
listeners, 1-3
Oracle Net Services
and load balancing, 4-13
and services, 4-4
Oracle Notification Service (ONS), 1-3
used by FAN, 4-4
Oracle Notification Services
API, 4-7
Oracle processes
managed by Oracle Clusterware, 1-3
Oracle RAC
cloning, 1-10, 7-1
software components, 1-5
Oracle RAC Management Processes (RMSn), 1-7
Oracle Real Application Clusters
and e-commerce, 10-6
installation overview, 1-9
overview of administration, 1-1
Oracle Scheduler
advanced job scheduling, xvii
Oracle Streams Advanced Queuing
and FAN, 4-8
ORACLE_BASE environment variable, 7-5, 7-8
ORACLE_HOME, 1-9
ORACLE_HOME environment variable, 7-5, 7-8
ORACLE_HOME_NAME environment variable, 7-5, 7-8
ORACLE_SID parameter, 3-14
OracleCRService resource
after cluster node restart, 3-5
OracleDBConsolesid
restarting after a cluster node restart, 3-5
OraMTS
external transaction manager, 4-23
OUI
database installation, 1-9
Oracle Real Application Clusters installation, 1-9
outages
planned, 4-9
unplanned, 4-8

P

parallel instance groups, 4-25
parallel processes, 4-25
parallel recovery, 6-6
PARALLEL_INSTANCE_GROUPS initialization parameter
parallel processes, 4-25
PARALLEL_MAX_SERVERS initialization parameter, 3-14
PARALLEL_MAX_SERVERS parameter, 6-6
parallelism
in Oracle Real Application Clusters, 10-7
parallel-aware query optimization, 10-7
parameter file
overview, 3-9
parameters
that must be identical on all instances, 3-13
that must be unique on all instances, 3-14
Partial ADDM mode, 11-7
performance, 11-3
aggregates by services, 11-4
comprehensive global data, xiv, 11-7
monitoring activity by wait events, services, and instances, 11-4
monitoring database throughput, 11-4
monitoring global cache block access, 11-3
monitoring potential problems in the database, 11-4
primary components affecting, 11-5
service aggregates by instances, 11-4
service aggregates by waits, 11-4
using ADDM, 1-13
performance evaluation
overview and concepts, 1-13
phases
cloning deployment, 7-3
cloning preparation, 7-2
PL/SQL
administering services, 4-28
PREFERRED instances
for services, 4-3
preferred read disks
ASM in an Oracle RAC extended distance cluster, xv, 2-5
private interconnect
determining usage, B-3
private network
alternative interconnect, 3-16
PRKA messages, C-2
PRKC messages, C-4
PRKD messages, C-14
PRKE messages, C-14
PRKH messages, C-15

PRKI messages, C-16
PRKN messages, C-18
PRKO messages, C-18
PRKP messages, C-22
PRKR messages, C-29
PRKS messages, C-35
PRKU messages, C-39
PRKV messages, C-39
processes
  managed by Oracle Clusterware, 1-3
  parallel, 4-25

Q

queue
  access using service names, 4-26
quiesce database
  in Oracle Real Application Clusters, 3-15
quiescing
  single-instance database, 3-15

R

raw device storage
  configuring when extending Oracle RAC, 9-6
raw devices, 1-4
rconfig command
  converting ASM single-instance, 2-6
rebalancing
  workloads, 4-2
RECOVER command, 3-4
  SQL*Plus, 3-4
recovery
  after SHUTDOWN ABORT, 3-5
  from multiple node failure, 6-3
  from single-node failure, 6-3
  instance, 3-5
  media failures, 6-5
  online, 6-3
  parallel, 6-6
  PARALLEL_MAX_SERVERS initialization parameter, 6-6
RECOVERY_PARALLELISM parameter, 6-6
redo log files
  instance recovery, 6-3
  log sequence number, 5-5
  using, 2-2
redo log groups, 2-2
redo logs
  format and destination specifications, 5-5
REMOTE_LOGIN_PASSWORD_FILE initialization parameter, 3-14
Resource Manager
  and services, 4-4
resource manager, 4-2
resource profiles
  and service creation, 4-4
resources
  releasing, 6-3
response file
  cloning, 7-6
restore scenarios
  RMAN, 6-1
restore scheme
  cluster file system, 6-2
  local file system, 6-2
RESULT_CACHE_MAX_SIZE initialization parameter, 3-13, 3-14
RMAN
  configuring channels, 5-3
  configuring channels to use automatic load balancing, 5-3
  configuring one channel for each instance, 5-3
  crosschecking on multiple nodes, 5-3
  local archiving scenario, 5-7
  restore scenarios, 6-1
RMSn Oracle RAC Management Processes, 1-7
rolling back
  instance recovery, 6-3
rolling upgrades
  ASM, xv
  ASM software, xv, 2-5
root.sh script
  $ORACLE_HOME, 7-8
  issues the oifcfg command during add node, 9-8
RSMN background process, 1-7
Runtime Connection Load Balancing, 4-4
runtime connection load balancing
  defined, 4-16
  disabling, 4-16
  in OCI session pools, 4-16
  introduction to, 4-2

S

scalability
  adding nodes and instances, 9-2
  Oracle RAC, 10-4
scheduler
  jobs, xvii
scripts
  $ORACLE_HOME/root.sh, 7-8
security
  wallet, 10-8
sequence numbers
  with Oracle Real Application Clusters, 10-5
sequences
  log sequence number, 5-5
Server Control
  see SRVCTL
Server Management
  administration of instances, 3-1
server parameter file
  backing up, 3-11
  setting values in, 3-10
server parameter files
  creating, 3-9, 3-11
servers
  scalability, 10-4
service

dependencies, 4-4
levels, 4-32
SRVCTL object noun name, A-5
service level objective
defining for Oracle RAC, 10-1
service metrics
OCI runtime connection load balancing, xiv, 4-16
runtime connection load balancing, 4-16
SERVICE parameter
FAN, and matching database signature, 4-9
SERVICE TIME
load balancing advisory goal, 4-11
SERVICE_NAMES initialization parameter, 3-13
setting for services, 3-13, 4-26
services
access to a queue, 4-26
activity levels aggregated by instances, 11-4
activity levels aggregated by services, 11-4
activity levels aggregated by waits, 11-4
administering, 4-24
administering with DBCA, 4-26
administering with Enterprise Manager, 4-26
administering with PL/SQL, 4-26
administering with PL/SQL DBMS_SERVICE package, 4-28
administering with SRVCTL, 4-26, 4-29
basic concepts about, 4-3
concepts, 1-8
configuring automatic workload management characteristics, 4-24
default, 4-4
introduction to, 1-4, 4-1
performance monitored by AWR, 4-4
SERVICE_NAMES parameter, 3-13, 4-26
specifying a service, 4-26
using, 4-3
sessions
multiple, 3-2
terminating on a specific instance, 3-7
SESSIONS_PER_USER initialization parameter, 3-13
setting attributes for SQLSetConnectAttr function, 4-18
setting instances, 1-12, 3-2
shared everything, 1-4
shared storage
configuring when extending Oracle RAC, 9-5
sharedhome argument, 9-14
SHOW INSTANCE command, 3-4
SQL*Plus, 3-4
SHOW PARAMETER command
SQL*Plus, 3-4
SHOW PARAMETERS command, 3-4
SHOW SGA command, 3-4
SQL*Plus, 3-4
SHUTDOWN ABORT command, 3-5
SHUTDOWN command
ABORT option, 3-5
SQL*Plus, 3-4
SHUTDOWN TRANSACTIONAL, 3-5
shutting down
instances, 3-5
shutting down an instance
abnormal shutdown, 3-5
shutting down instances, 3-5
sidalrt.log file, B-2
single system image, 1-5
single-instance ASM
converting to clustered ASM, 2-6
single-instance databases
quiescing, 3-15
SMON process
instance recovery, 6-3
recovery after SHUTDOWN ABORT, 3-5
snapshot control file, 5-1
SPFILE
restore with Enterprise Manager, 6-2
restore with RMAN, 6-2
SPFILE initialization parameter, 3-13, 3-15
SQL statements
executing in parallel, 4-25
instance-specific, 3-3
SQL*Plus, 3-2
changing the prompt, 3-3
SQL*Plus commands
effect on instances, 3-3
SQL_ORCLATTR_FAILOVER_CALLBACK attribute
of the SQLSetConnectAttr function, 4-18
SQL_ORCLATTR_FAILOVER_HANDLE attribute
of the SQLSetConnectAttr function, 4-18
SQLSetConnectAttr function
setting attributes in the sqora.h file, 4-18
SQL_ORCLATTR_FAILOVER_CALLBACK attribute, 4-18
SQL_ORCLATTR_FAILOVER_HANDLE attribute, 4-18
sqora.h file, 4-18
SRVCTL
add asm command, A-8
add database command, A-6
add instance command, A-6
add listener command
listeners
adding to a node, A-8, A-9
add nodeapps command, A-8
add service command, A-8
add, usage description, A-5
administering ASM instances, 2-9
administering services with, 4-29
cluster database configuration tasks, A-3
cluster database tasks, A-3
concurrent commands, A-2
config asm command, A-11
config database command, A-9
config nodeapps command, A-10
config service command, A-10
config, usage description, A-9
disable asm command, A-15
disable database command, A-13
disable instance command, A-14
disable service command, 4-30, A-14

disable, usage description, A-13
enable asm command, A-13
enable database command, A-11
enable instance command, A-12
enable service command, 4-30, A-12
enable, usage description, A-11
getenv database command, A-28
getenv instance command, A-29
getenv nodeapps command, A-29
getenv service command, A-29
getenv, usage description, A-28
modify database command, A-21
modify instance command, A-22
modify listener command, A-25
modify nodeapps command, A-24
modify service command, A-23
modify, usage description, A-21
node-level tasks, A-3
object noun names, A-5
overview, 3-4
overview and concepts, 1-12
relocate service command, A-25
relocate, usage description, A-25
remove asm command, A-36
remove database command, A-34
remove instance command, A-34
remove nodeapps command, A-35
remove service command, A-35
remove, usage description, A-33
setenv database command, A-30
setenv instance command, A-31
setenv nodeapps command, A-32
setenv service command, A-31
setenv, usage description, A-30
start asm command, A-17
start database command, 3-7, A-15
start instance command, 3-6, A-16
start listener command, A-17
start nodeapps command, A-17
start service command, 4-30, A-16
start, usage description, A-15
status asm command, A-28
status database command, A-26
status instance command, A-27
status nodeapps command, A-27
status service command, 4-30, A-27
status, usage description, A-26
stop asm command, A-20
stop database command, 3-7, A-18
stop instance command, 3-6, A-19
stop listener command, A-21
stop nodeapps command, A-20
stop service command, A-19
stop, usage description, A-18
unsetenv database command, A-32
unsetenv instance command, A-32
unsetenv nodeapps command, A-33
unsetenv service command, A-33
unsetenv, usage description, A-30
srvctl remove listener command, A-36
SRVM_TRACE environment variable, B-3
Standard Edition
installing ASM, 1-9
START_SERVICE procedure
DBMS_SERVICE PL/SQL package, 4-29
STARTUP command
SQL*Plus, 3-4
statistics
contents of, 11-6
monitoring for Oracle Real Application Clusters, 11-6
Statspack, 11-8, 11-9
usage, 1-13
STOP_SERVICE procedure
DBMS_SERVICE PL/SQL package, 4-29
storage
Oracle Real Application Clusters
introduction, 2-4
storage
cluster file system, 2-1
storage devices
configuring when extending Oracle RAC, 9-5
subnet
configuring for virtual IP address, A-3
Summary dialog, 9-10
SYS$BACKGROUND service, 4-5
SYS$USERS service, 4-5
SYSASM privilege, 3-6
SYSAUX tablespace, 10-6
SYSDBA
privilege for connecting, 3-3
SYSOPER privilege
for connecting, 3-3
system change, 5-5
System Global Area (SGA), 1-6
size requirements, 1-6

T

tablespaces
automatic segment-space management in Oracle RAC, 10-5
automatic undo management in Oracle RAC, 10-5
use in Oracle RAC, 10-5
TCP/IP, 1-5
THREAD initialization parameter, 3-13
threads
multiple application, 4-16
thresholds
services, 4-32
THROUGHPUT
load balancing advisory goal, 4-11
timed statistics, 11-6
timeout
messages, avoiding, 1-5
Top Activity drill down menu
on the Cluster Database Performance page, 11-4
Top Cluster Events, ASH report, 11-8

Top Remote Instance, ASH report, 11-8
trace files, B-2
  managing, B-1
  sidalrt.log, B-2
TRACE_ENABLED initialization parameter, 3-15
tracing
  enabling Java tools, B-3
  SRVM_TRACE environment variable, B-3
  writing to log files, B-3
transactions
  instance failure, 6-3
  rolling back, 6-3
  waiting for recovery, 6-3
Transparent Application Failover
  and services, 4-6
Transparent Data Encryption, 10-8
  specifying the ENCRYPTION_WALLET_
    LOCATION parameter, 10-8
  specifying the WALLET_LOCATION
    parameter, 10-8
Transparent Database Encryption, 10-8
tuning
  overview, 1-13
  using ADDM, 1-13

U

UNDO_MANAGEMENT initialization
    parameter, 3-14
UNDO_RETENTION initialization parameter, 3-15
UNDO_TABLESPACE parameter, 3-14
upgrades
  ASM rolling upgrades, xv, 2-5

V

V$ACTIVE_INSTANCES, 3-7
vendor cluster file systems
  configuring when extending Oracle RAC, 9-6
vendor clusterware, 1-1
verification
  datafiles, online files, 2-2
versions
  compatibility for Oracle RAC and Oracle Database
    software, 1-9
views
  creating for Oracle Real Application
    Clusters, 11-6
  for performance monitoring, 11-6
VIP addresses, 1-5
  configuring, 1-5
virtual host name, 1-5
Virtual Internet Protocol (VIP) address, 1-3
virtual IP address
  requirements, A-3
volume manager
  alternate storage method for Oracle RAC, 1-4

W

wait events, 11-9
  aggregated for service performance, 11-4
wallet
  data security, 10-8
WALLET_LOCATION parameter, 10-8
wallets
  for Transparent Data Encryption, 10-8
Welcome page, 9-9
workload
  balancing with Oracle Scheduler, xvii
workload management
  See automatic workload management
workloads
  and services, 4-3
  See Also automatic workload management

X

XA transactions, 4-22, 10-6
  spanning Oracle RAC instances, xiv, 4-22