

Oracle® Real Application Clusters Administration and Deployment Guide



18c
E83996-02
February 2018

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

ORACLE®

Oracle Real Application Clusters Administration and Deployment Guide, 18c

E83996-02

Copyright © 1999, 2018, Oracle and/or its affiliates. All rights reserved.

Primary Author: Richard Strohm

Contributing Authors: Janet Stern

Contributors: Troy Anthony, Lance Ashdown, Ram Avudaiappan, Prasad Bagal, Mark Bauer, Anand Beldalker, Eric Belden, Gajanan Bhat, David Brower, George Claborn, Maria Colgan, Carol Colrain, Jonathan Creighton, Rajesh Dasari, Mark Dilman, Richard Frank, GP Prabhaker Gongloor, Wei Hu, Yong Hu, Dominique Jeunot, Sameer Joshi, Raj K. Kammend, Ankita Khandelwal, Sana Karam, Roland Knapp, Karen Li, Barb Lundhild, Venkat Maddali, Bill Manry, John McHugh, Saar Maoz, Matthew Mckerley, Markus Michalewicz, Anil Nair, Philip Newlan, Michael Nowak, Muthu Olagappan, Bharat Paliwal, Hanlin Qian, Hairong Qin, Mark Ramacher, Sampath Ravindhran, Kevin Reardon, Kathy Rich, Dipak Saggi, Daniel Semler, Ara Shakian, Cathy Shea, Khethavath P. Singh, Kesavan Srinivasan, Leo Tominna, Peter Wahl, Tak Wang, Richard Wessman, Douglas Williams, Mike Zampiceni, Michael Zoll

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

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

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

Preface

Audience	xxii
Documentation Accessibility	xxii
Related Documents	xxiii
Conventions	xxiii

Changes in This Release for Oracle Real Application Clusters Administration and Deployment Guide

Changes in Oracle Real Application Clusters Release 18c, Version 18.1	xxv
Changes in Oracle Real Application Clusters 12c Release 2 (12.2)	xxvi
Changes in Oracle Real Application Clusters 12c Release 1 (12.1)	xxviii

1 Introduction to Oracle RAC

Overview of Oracle RAC	1-1
Overview of Installing Oracle RAC	1-3
Understanding Compatibility in Oracle RAC Environments	1-4
Oracle RAC Database Management Styles and Database Installation	1-4
Oracle RAC Database Management Styles and Database Creation	1-5
Overview of Extending an Oracle RAC Cluster	1-6
Overview of Oracle Real Application Clusters One Node	1-7
Overview of Oracle Clusterware for Oracle RAC	1-8
Overview of Oracle Flex Clusters	1-9
Overview of Reader Nodes	1-9
Overview of Local Temporary Tablespaces	1-10
Overview of Oracle RAC Architecture and Processing	1-16
Understanding Cluster-Aware Storage Solutions	1-17
Oracle RAC and Network Connectivity	1-17
Overview of Using Dynamic Database Services to Connect to Oracle Databases	1-18
Overview of Virtual IP Addresses	1-18
Restricted Service Registration in Oracle RAC	1-19

About Oracle RAC Software Components	1-19
About Oracle RAC Background Processes	1-20
Overview of Automatic Workload Management with Dynamic Database Services	1-21
Overview of Server Pools and Policy-Managed Databases	1-24
Introduction to Server Pools	1-24
Examples of Using Server Pools	1-24
Deploying Policy-Managed Databases	1-28
Managing Policy-Managed Databases	1-28
Policy-Based Cluster Management	1-29
Overview of Oracle Database Quality of Service Management	1-30
Overview of Hang Manager	1-31
Overview of Oracle Multitenant with Oracle RAC	1-32
Overview of Database In-Memory and Oracle RAC	1-32
Overview of Managing Oracle RAC Environments	1-33
About Designing and Deploying Oracle RAC Environments	1-33
About Administrative Tools for Oracle RAC Environments	1-33
About Monitoring Oracle RAC Environments	1-34
About Evaluating Performance in Oracle RAC Environments	1-35

2 Administering Storage in Oracle RAC

Overview of Storage Management for Oracle RAC	2-2
Data File Access in Oracle RAC	2-2
NFS Server for Storage	2-3
Redo Log File Storage in Oracle RAC	2-3
Automatic Undo Management in Oracle RAC	2-4
Oracle Automatic Storage Management with Oracle RAC	2-4
Storage Management in Oracle RAC	2-5
Modifying Disk Group Configurations for Oracle ASM	2-5
Oracle ASM Disk Group Management	2-6
Configuring Preferred Mirror Read Disks in Extended Distance Clusters	2-6
Converting Nonclustered Oracle ASM to Clustered Oracle ASM	2-6
Administering Oracle ASM Instances with SRVCTL in Oracle RAC	2-6

3 Administering Database Instances and Cluster Databases

Overview of Oracle RAC Database Administration	3-1
Tools for Administering Oracle RAC	3-4
Administering Oracle RAC with SRVCTL	3-4
Administering Oracle RAC with Oracle Enterprise Manager	3-4
Administering Oracle RAC with SQL*Plus	3-5

Starting and Stopping Instances and Oracle RAC Databases	3-7
Starting One or More Instances and Oracle RAC Databases Using SRVCTL	3-8
Stopping One or More Instances and Oracle RAC Databases Using SRVCTL	3-9
Stopping All Databases and Instances Using CRSCTL	3-9
Starting and Stopping Individual Instances Using SQL*Plus	3-10
Starting and Stopping PDBs in Oracle RAC	3-11
Verifying That Instances are Running	3-12
Using SRVCTL to Verify That Instances are Running	3-12
Using SQL*Plus to Verify That Instances are Running	3-12
Terminating Sessions On a Specific Cluster Instance	3-13
Overview of Initialization Parameter Files in Oracle RAC	3-15
Setting SPFILE Parameter Values for Oracle RAC	3-16
Parameter File Search Order in Oracle RAC	3-17
Backing Up the Server Parameter File	3-17
Initialization Parameter Use in Oracle RAC	3-18
Parameters That Must Have Identical Settings on All Instances	3-20
Parameters That Have Unique Settings on All Instances	3-21
Parameters That Should Have Identical Settings on All Instances	3-22
Converting an Administrator-Managed Database to a Policy-Managed Database	3-23
Managing Memory Pressure for Database Servers	3-24
Quiescing Oracle RAC Databases	3-25
Administering Multiple Cluster Interconnects on Linux and UNIX Platforms	3-26
Use Cases for Setting the CLUSTER_INTERCONNECTS Parameter	3-26
Customizing How Oracle Clusterware Manages Oracle RAC Databases	3-28
Advanced Oracle Enterprise Manager Administration	3-29
Using Oracle Enterprise Manager Cloud Control to Discover Nodes and Instances	3-29
Other Oracle Enterprise Manager Capabilities	3-30
Administering Jobs and Alerts in Oracle RAC	3-31
Administering Jobs in Oracle RAC	3-31
Administering Alerts in Oracle RAC with Oracle Enterprise Manager	3-31
Using Defined Blackouts in Oracle Enterprise Manager	3-32

4 Administering Oracle RAC One Node

Creating an Oracle RAC One Node Database	4-1
Converting Databases	4-2
Converting a Database from Oracle RAC to Oracle RAC One Node	4-2
Converting a Database from Oracle RAC One Node to Oracle RAC	4-3
Online Database Relocation	4-3

5 Workload Management with Dynamic Database Services

Connection Load Balancing	5-1
Server-Side Load Balancing	5-2
Generic Database Clients	5-2
Client-Side Connection Configuration for Older Clients	5-3
JDBC-Thin Clients	5-3
OCI Clients	5-4
Client-Side Load Balancing	5-4
Load Balancing Advisory	5-5
Overview of the Load Balancing Advisory	5-5
Configuring Your Environment to Use the Load Balancing Advisory	5-5
Load Balancing Advisory FAN Events	5-6
Monitoring Load Balancing Advisory FAN Events	5-7
Enabling Clients for Oracle RAC	5-8
Overview of Oracle Integrated Clients and FAN	5-8
Enabling JDBC-Thin Clients for Fast Connection Failover	5-9
Oracle Notification Service for JDBC-Thin Clients	5-9
Configuring FCF for JDBC/OCI and JDBC-Thin Driver Clients	5-10
Enabling JDBC Clients for Run-time Connection Load Balancing	5-11
Configuring JDBC-Thin Clients for Application Continuity for Java	5-11
Configuring JDBC-Thin Clients for Transaction Guard	5-12
Enabling OCI Clients for Fast Connection Failover	5-12
Enabling OCI Clients for Run-time Connection Load Balancing	5-13
Configuring OCI Clients to use Transaction Guard	5-14
Enabling ODP.NET Clients to Receive FAN High Availability Events	5-14
Enabling ODP.NET Clients to Receive FAN Load Balancing Advisory Events	5-15
Configuring ODP.NET Clients to use Transaction Guard	5-15
Distributed Transaction Processing in Oracle RAC	5-16
Overview of XA Transactions and Oracle RAC	5-16
Using Global Transactions and XA Affinity for XA Transactions	5-17
Using Services with XA Transactions on Oracle RAC	5-18
Configuring Services for XA Applications	5-18
Relocating Services in Administrator-Managed Databases	5-19
Oracle RAC Sharding	5-19
Automatic Workload Repository	5-20
Measuring Performance by Service Using the Automatic Workload Repository	5-21
Automatic Workload Repository Service Thresholds and Alerts	5-22
Example of Services and Thresholds Alerts	5-23
Enable Service, Module, and Action Monitoring	5-23
Using Oracle Services	5-24

Service Deployment Options	5-24
Service Usage in an Oracle RAC Database	5-24
Oracle Clusterware Resources for a Service	5-25
Database Resource Manager Consumer Group Mappings for Services	5-25
Performance Monitoring by Service with AWR	5-25
Parallel Operations and Services	5-25
Oracle Streams and Oracle RAC	5-25
Service Characteristics	5-26
Service Name	5-26
Service Edition	5-26
Service Management Policy	5-26
Database Role for a Service	5-27
Instance Preference	5-27
Server Pool Assignment	5-28
Load Balancing Advisory Goal for Run-time Connection Load Balancing	5-28
Connection Load Balancing Goal	5-29
Distributed Transaction Processing	5-29
Default Service Connections	5-29
Restricted Service Registration	5-30
Administering Services	5-30
Overview of Service Administration	5-31
Administering Services with Oracle Enterprise Manager	5-32
Administering Services with SRVCTL	5-33
Creating Services with SRVCTL	5-34
Creating Services for Application Continuity and Transaction Guard	5-34
Starting and Stopping Services with SRVCTL	5-35
Enabling and Disabling Services with SRVCTL	5-36
Relocating Services with SRVCTL	5-36
Obtaining the Status of Services with SRVCTL	5-36
Obtaining the Configuration of Services with SRVCTL	5-36
Global Services	5-37
Service-Oriented Buffer Cache Access	5-38
Connecting to a Service: An Example	5-38

6 Ensuring Application Continuity

Fast Application Notification	6-2
Overview of Fast Application Notification	6-2
Fast Application Notification High Availability Events	6-6
Subscription to High Availability Events	6-8
Using Fast Application Notification Callouts	6-8

Managing Unplanned Outages	6-9
Managing Planned Maintenance	6-10
Managing Planned Maintenance Without User Interruption	6-11
Managing a Group of Services for Maintenance	6-13
Starting Services	6-14
Pluggable Database-Level Operations	6-14
Relocating Services	6-15
Stopping Services	6-16
Server Draining Ahead of Planned Maintenance	6-16
About Application Continuity	6-21
Key Concepts for Application Continuity	6-21
Transparent Application Continuity	6-23
Transparent Application Continuity for Various Applications	6-24
Application Continuity Operation and Usage	6-26
How Application Continuity Works for Applications	6-26
Actions for Using Application Continuity	6-28
Configuring Oracle Application Continuity	6-32
Configuring Oracle Database for Application Continuity	6-33
Establishing the Initial State Before Application Continuity Replays	6-34
Delaying the Reconnection in Application Continuity	6-37
Using Application Continuity with SQL*Plus	6-39
Using Application Continuity for Planned Maintenance	6-39
Running Without Application Continuity	6-40
Disabling Replay in Application Continuity	6-40
Terminating or Disconnecting a Session Without Replay	6-42
Mutable Functions and Application Continuity	6-43
Administering Mutable Values	6-45
Protection-Level Statistics	6-46
Session State Consistency	6-46
Auto Session State Consistency	6-47
Dynamic Session State Consistency	6-47
Static Session State Consistency	6-49
Potential Side Effects of Application Continuity	6-50
Restrictions and Other Considerations for Application Continuity	6-51
Transaction Guard for Improving Client Failover	6-53
Transaction Guard Configuration Checklist	6-54
Configuring Services for Transaction Guard	6-54
Failing Over OCI Clients with TAF	6-55

7 Configuring Recovery Manager and Archiving

Overview of Configuring RMAN for Oracle RAC	7-1
Archiving Mode in Oracle RAC	7-1
Configuring the RMAN Snapshot Control File Location	7-2
Configuring RMAN to Automatically Backup the Control File and SPFILE	7-2
Crosschecking on Multiple Oracle RAC Nodes	7-3
Configuring Channels for RMAN in Oracle RAC	7-3
Configuring Channels to Use Automatic Load Balancing	7-4
Configuring Channels to Use a Specific Node	7-4
Managing Archived Redo Logs Using RMAN in Oracle RAC	7-4
Archived Redo Log File Conventions in Oracle RAC	7-5
RMAN Archiving Configuration Scenarios	7-5
Oracle Automatic Storage Management and Cluster File System Archiving Scheme	7-6
Advantages of the Cluster File System Archiving Scheme	7-6
Initialization Parameter Settings for the Cluster File System Archiving Scheme	7-7
Location of Archived Logs for the Cluster File System Archiving Scheme	7-7
Noncluster File System Local Archiving Scheme	7-7
Considerations for Using Noncluster File System Local Archiving	7-7
Initialization Parameter Settings for Noncluster File System Local Archiving	7-8
Location of Archived Logs for Noncluster File System Local Archiving	7-8
File System Configuration for Noncluster File System Local Archiving	7-9
Monitoring the Archiver Processes	7-10

8 Managing Backup and Recovery

RMAN Backup Scenario for Noncluster File System Backups	8-1
RMAN Restore Scenarios for Oracle RAC	8-1
Restoring Backups from a Cluster File System	8-2
Restoring Backups from a Noncluster File System	8-2
Using RMAN or Oracle Enterprise Manager to Restore the Server Parameter File (SPFILE)	8-3
Instance Recovery in Oracle RAC	8-3
Single Node Failure in Oracle RAC	8-3
Multiple-Node Failures in Oracle RAC	8-4
Using RMAN to Create Backups in Oracle RAC	8-4
Channel Connections to Cluster Instances with RMAN	8-4
Node Affinity Awareness of Fast Connections	8-5
Deleting Archived Redo Logs after a Successful Backup	8-5
Autolocation for Backup and Restore Commands	8-6

Media Recovery in Oracle RAC	8-6
Parallel Recovery in Oracle RAC	8-6
Parallel Recovery with RMAN	8-6
Disabling Parallel Recovery	8-7
Disabling Instance and Crash Recovery Parallelism	8-7
Disabling Media Recovery Parallelism	8-7
Using a Fast Recovery Area in Oracle RAC	8-7

9 Cloning Oracle RAC to Nodes in a New Cluster

Introduction to Cloning Oracle RAC	9-1
Preparing to Clone Oracle RAC	9-2
Deploying Oracle RAC Clone to Nodes in a Cluster	9-3
Locating and Viewing Log Files Generated During Cloning	9-6

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

About Adding Nodes Using Cloning in Oracle RAC Environments	10-1
Cloning Local Oracle Homes on Linux and UNIX Systems	10-1
Cloning Shared Oracle Homes on Linux and UNIX Systems	10-2
Cloning Oracle Homes on Windows Systems	10-3

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

Adding Oracle RAC to Nodes with Oracle Clusterware Installed	11-1
Adding Policy-Managed Oracle RAC Database Instances to Target Nodes	11-3
Adding Administrator-Managed Oracle RAC Database Instances to Target Nodes	11-3
Using DBCA in Interactive Mode to Add Database Instances to Target Nodes	11-4
Using DBCA in Silent Mode to Add Database Instances to Target Nodes	11-5
Deleting Oracle RAC from a Cluster Node	11-5
Deleting Instances from Oracle RAC Databases	11-6
Using DBCA in Interactive Mode to Delete Instances from Nodes	11-6
Using DBCA in Silent Mode to Delete Instances from Nodes	11-7
Removing Oracle RAC	11-8
Deleting Nodes from the Cluster	11-8

12 Adding and Deleting Oracle RAC from Nodes on Windows Systems

Adding Oracle RAC to Nodes with Oracle Clusterware Installed	12-1
Adding Administrator-Managed Oracle RAC Database Instances to Target Nodes	12-3
Using DBCA in Interactive Mode to Add Database Instances to Target Nodes	12-4
Using DBCA in Silent Mode to Add Database Instances to Target Nodes	12-5
Deleting Oracle RAC from a Cluster Node	12-5
Deleting Instances from Oracle RAC Databases	12-6
Using DBCA in Interactive Mode to Delete Instances from Nodes	12-6
Using DBCA in Silent Mode to Delete Instances from Nodes	12-7
Removing Oracle RAC	12-8
Deleting Nodes from the Cluster	12-8

13 Design and Deployment Techniques

Deploying Oracle RAC for High Availability	13-1
About Designing a High Availability System	13-1
Best Practices for Deploying Oracle RAC in a High Availability Environment	13-2
Consolidating Multiple Applications in a Database or Multiple Databases in a Cluster	13-3
Managing Capacity During Consolidation	13-3
Managing the Global Cache Service Processes During Consolidation	13-4
Using a Database Cloud for Consolidation	13-4
Scalability of Oracle RAC	13-5
General Design Considerations for Oracle RAC	13-5
General Database Deployment Topics for Oracle RAC	13-6
Tablespace Use in Oracle RAC	13-6
Object Creation and Performance in Oracle RAC	13-6
Node Addition and Deletion and the SYSAUX Tablespace in Oracle RAC	13-7
Distributed Transactions and Oracle RAC	13-7
Deploying OLTP Applications in Oracle RAC	13-7
Flexible Implementation with Cache Fusion	13-7
Deploying Data Warehouse Applications with Oracle RAC	13-8
Speed-Up for Data Warehouse Applications on Oracle RAC	13-8
Parallel Execution in Data Warehouse Systems and Oracle RAC	13-8
Data Security Considerations in Oracle RAC	13-8
Transparent Data Encryption and Keystores	13-8
Windows Firewall Considerations	13-10

14 Monitoring Performance

Overview of Monitoring and Tuning Oracle RAC Databases	14-1
Monitoring Oracle RAC and Oracle Clusterware	14-1
The Cluster Database Home Page	14-2
The Interconnects Page	14-2
The Cluster Database Performance Page	14-3
Tuning Oracle RAC Databases	14-4
Verifying the Interconnect Settings for Oracle RAC	14-4
Influencing Interconnect Processing	14-5
Performance Views in Oracle RAC	14-5
Creating Oracle RAC Data Dictionary Views with CATCLUST.SQL	14-6
Oracle RAC Performance Statistics	14-6
Automatic Workload Repository in Oracle RAC Environments	14-6
Active Session History Reports for Oracle RAC	14-7
Overview of ASH Reports for Oracle RAC	14-7
ASH Report for Oracle RAC: Top Cluster Events	14-7
ASH Report for Oracle RAC: Top Remote Instance	14-7
Monitoring Oracle RAC Statistics and Wait Events	14-8
Oracle RAC Statistics and Events in AWR and Statspack Reports	14-8
Oracle RAC Wait Events	14-8
Monitoring Performance by Analyzing GCS and GES Statistics	14-9
Analyzing the Effect of Cache Fusion in Oracle RAC	14-9
Analyzing Performance Using GCS and GES Statistics	14-9
Analyzing Cache Fusion Transfer Impact Using GCS Statistics	14-10
Analyzing Response Times Based on Wait Events	14-11
Block-Related Wait Events	14-11
Message-Related Wait Events	14-11
Contention-Related Wait Events	14-12
Load-Related Wait Events	14-12

15 Converting Single-Instance Oracle Databases to Oracle RAC and Oracle RAC One Node

Administrative Issues for Converting Databases to Oracle RAC	15-1
Converting to Oracle RAC and Oracle RAC One Node Using DBCA	15-2
Converting Oracle Database Installations to Oracle RAC Using DBCA	15-2
Use DBCA to Create an Image of the Single-Instance Database	15-3
Complete the Oracle Clusterware Installation	15-3
Validate the Cluster	15-3
Copy the Preconfigured Database Image	15-3

Install Oracle Database 12c Software with Oracle RAC	15-4
Converting Single Instance on a Cluster to Oracle RAC One Node Using DBCA	15-4
Converting Single Instance on a Cluster to Oracle RAC Using DBCA	15-4
Single-Instance Database on a Cluster Running from an Oracle RAC-Enabled Home	15-5
Single-Instance Database on a Cluster Running from an Oracle RAC-Disabled Home	15-7
Preparing to Convert with rconfig and Oracle Enterprise Manager	15-7
Prerequisites for Converting to Oracle RAC Databases	15-8
Configuration Changes During Oracle RAC Conversion Using rconfig	15-8
Converting Databases to Oracle RAC Using rconfig or Oracle Enterprise Manager	15-9
Converting Databases to Oracle RAC Using Oracle Enterprise Manager	15-10
Converting Databases to Oracle RAC Using rconfig	15-11
Example of rconfig XML Input Files for ConvertToRAC	15-12
Postconversion Steps	15-14

A Server Control Utility Reference

SRVCTL Usage Information	A-1
Specifying Command Parameters as Keywords Instead of Single Letters	A-3
Character Set and Case Sensitivity of SRVCTL Object Values	A-3
Summary of Tasks for Which SRVCTL Is Used	A-4
Using SRVCTL Help	A-6
SRVCTL Privileges and Security	A-7
Additional SRVCTL Topics	A-7
Deprecated SRVCTL Subprograms or Commands	A-7
Single Character Parameters for all SRVCTL Commands	A-8
Miscellaneous SRVCTL Commands and Parameters	A-15
SRVCTL Command Reference	A-15
svctl add database	A-17
svctl config database	A-20
svctl convert database	A-21
svctl disable database	A-22
svctl downgrade database	A-23
svctl enable database	A-23
svctl getenv database	A-24
svctl modify database	A-24
svctl predict database	A-28
svctl relocate database	A-28
svctl remove database	A-29
svctl setenv database	A-30

svctl start database	A-31
svctl status database	A-32
svctl stop database	A-33
svctl unsetenv database	A-34
svctl update database	A-35
svctl upgrade database	A-35
svctl disable diskgroup	A-35
svctl enable diskgroup	A-36
svctl predict diskgroup	A-36
svctl remove diskgroup	A-37
svctl start diskgroup	A-37
svctl status diskgroup	A-38
svctl stop diskgroup	A-38
svctl start home	A-39
svctl status home	A-39
svctl stop home	A-40
svctl add instance	A-41
svctl disable instance	A-42
svctl enable instance	A-42
svctl modify instance	A-43
svctl remove instance	A-44
svctl start instance	A-45
svctl status instance	A-46
svctl stop instance	A-46
svctl update instance	A-47
svctl add listener	A-48
svctl config listener	A-50
svctl disable listener	A-51
svctl enable listener	A-51
svctl getenv listener	A-52
svctl modify listener	A-52
svctl predict listener	A-53
svctl remove listener	A-54
svctl setenv listener	A-54
svctl start listener	A-55
svctl status listener	A-55
svctl stop listener	A-56
svctl unsetenv listener	A-56
svctl update listener	A-57
svctl add network	A-57
svctl config network	A-58

svctl modify network	A-59
svctl predict network	A-60
svctl remove network	A-60
svctl add nodeapps	A-61
svctl config nodeapps	A-62
svctl disable nodeapps	A-63
svctl enable nodeapps	A-63
svctl getenv nodeapps	A-64
svctl modify nodeapps	A-65
svctl remove nodeapps	A-66
svctl setenv nodeapps	A-67
svctl start nodeapps	A-68
svctl status nodeapps	A-68
svctl stop nodeapps	A-68
svctl unsetenv nodeapps	A-69
svctl add ons	A-70
svctl config ons	A-71
svctl disable ons	A-71
svctl enable ons	A-71
svctl modify ons	A-71
svctl remove ons	A-72
svctl start ons	A-72
svctl status ons	A-73
svctl stop ons	A-73
svctl add scan	A-73
svctl config scan	A-74
svctl disable scan	A-75
svctl enable scan	A-75
svctl modify scan	A-75
svctl predict scan	A-76
svctl relocate scan	A-76
svctl remove scan	A-77
svctl start scan	A-78
svctl status scan	A-78
svctl stop scan	A-79
svctl add scan_listener	A-79
svctl config scan_listener	A-80
svctl disable scan_listener	A-81
svctl enable scan_listener	A-81
svctl modify scan_listener	A-82
svctl predict scan_listener	A-82

svctl relocate scan_listener	A-83
svctl remove scan_listener	A-83
svctl start scan_listener	A-84
svctl status scan_listener	A-84
svctl stop scan_listener	A-85
svctl update scan_listener	A-85
svctl relocate server	A-85
svctl status server	A-86
svctl add service	A-86
svctl config service	A-92
svctl disable service	A-94
svctl enable service	A-95
svctl modify service	A-96
svctl predict service	A-102
svctl relocate service	A-102
svctl remove service	A-104
svctl start service	A-105
svctl status service	A-107
svctl stop service	A-107
svctl add srvpool	A-109
svctl config srvpool	A-110
svctl modify srvpool	A-111
svctl remove srvpool	A-112
svctl status srvpool	A-112
svctl add vip	A-113
svctl config vip	A-114
svctl disable vip	A-114
svctl enable vip	A-114
svctl getenv vip	A-115
svctl modify vip	A-115
svctl predict vip	A-116
svctl relocate vip	A-116
svctl remove vip	A-117
svctl setenv vip	A-118
svctl start vip	A-118
svctl status vip	A-119
svctl stop vip	A-119
svctl unsetenv vip	A-120
svctl config volume	A-121
svctl disable volume	A-122
svctl enable volume	A-123

svctl remove volume	A-123
svctl start volume	A-124
svctl status volume	A-125
svctl stop volume	A-126

B Troubleshooting Oracle RAC

Where to Find Files for Analyzing Errors	B-1
Managing Diagnostic Data in Oracle RAC	B-2
Using Instance-Specific Alert Files in Oracle RAC	B-3
Enabling Tracing for Java-Based Tools and Utilities in Oracle RAC	B-3
Resolving Pending Shutdown Issues	B-3
How to Determine If Oracle RAC Instances Are Using the Private Network	B-3
Database Fails to Start after Private NIC Failure	B-4

Glossary

Index

List of Tables

3-1	How SQL*Plus Commands Affect Instances	3-6
3-2	Descriptions of V\$ACTIVE_INSTANCES Columns	3-13
3-3	Initialization Parameters Specific to Oracle RAC	3-18
3-4	Parameters That Should Have Identical Settings on All Instances	3-22
5-1	Load Balancing Advisory FAN Events	5-6
6-1	Event Parameter Name-Value Pairs and Descriptions	6-6
6-2	FAN Parameters and Matching Session Information	6-8
6-3	Standard Connection Tests for Some Common Application Servers	6-20
6-4	Example Treatment of Mutable Objects by Products During Replay	6-43
7-1	Archived Redo Log File Name Format Parameters	7-5
7-2	UNIX/NFS Location Log Examples, Noncluster File System Local Archiving	7-9
7-3	UNIX/NFS Configuration for Shared Read Local Archiving Examples	7-9
9-1	clone.pl Script Parameters	9-4
9-2	Environment Variables Passed to the clone.pl Script	9-5
9-3	Cloning Parameters Passed to the clone.pl Script.	9-5
9-4	Finding the Location of the Oracle Inventory Directory	9-7
11-1	Variables in the DBCA Silent Mode Syntax	11-5
A-1	String Restrictions for SRVCTL Object Names	A-3
A-2	Deprecated Single-Character Parameters for SRVCTL Commands	A-8
A-3	Deprecated Commands and Parameters for SRVCTL	A-15
A-4	Object Keywords and Abbreviations	A-16
A-5	srvctl add database Command Parameters	A-18
A-6	srvctl config database Command Parameters	A-21
A-7	srvctl convert database Command Parameters	A-22
A-8	srvctl disable database Command Parameters	A-23
A-9	srvctl downgrade database Command Parameters	A-23
A-10	srvctl enable database Command Parameters	A-24
A-11	srvctl getenv database Command Parameters	A-24
A-12	srvctl modify database Command Parameters	A-25
A-13	srvctl relocate database Command Parameters	A-28
A-14	srvctl remove database Command Parameters	A-30
A-15	srvctl setenv database Command Parameters	A-30
A-16	srvctl start database Command Parameters	A-31
A-17	srvctl status database Parameters	A-32
A-18	srvctl stop database Command Parameters	A-34

A-19	svctl unsetenv database Command Parameters	A-35
A-20	svctl upgrade database Command Parameters	A-35
A-21	svctl disable diskgroup Command Parameters	A-36
A-22	svctl enable diskgroup Command Parameters	A-36
A-23	svctl start diskgroup Command Parameters	A-37
A-24	svctl status diskgroup Command Parameters	A-38
A-25	svctl stop diskgroup Command Parameters	A-38
A-26	svctl start home Command Parameters	A-39
A-27	svctl status home Command Parameters	A-40
A-28	svctl stop home Command Parameters	A-40
A-29	svctl add instance Command Parameters	A-41
A-30	svctl disable instance Command Parameters	A-42
A-31	svctl enable instance Command Parameters	A-43
A-32	svctl modify instance Command Parameters	A-43
A-33	svctl remove instance Command Parameters	A-44
A-34	svctl start instance Parameters	A-45
A-35	svctl stop instance Command Parameters	A-46
A-36	svctl add listener Command Parameters	A-48
A-37	svctl config listener Command Parameters	A-50
A-38	svctl disable listener Command Parameters	A-51
A-39	svctl enable listener Command Parameters	A-51
A-40	svctl getenv listener Command Parameters	A-52
A-41	svctl modify listener Command Parameters	A-53
A-42	svctl setenv listener Command Parameters	A-54
A-43	svctl start listener Command Parameters	A-55
A-44	svctl status listener Command Parameters	A-55
A-45	svctl stop listener Command Parameters	A-56
A-46	svctl unsetenv listener Command Parameters	A-57
A-47	svctl add network Command Parameters	A-57
A-48	svctl modify network Command Parameters	A-59
A-49	svctl remove network Command Parameters	A-61
A-50	svctl add nodeapps Command Parameters	A-62
A-51	svctl disable nodeapps Command Parameters	A-63
A-52	svctl enable nodeapps Command Parameters	A-64
A-53	svctl getenv nodeapps Command Parameters	A-64
A-54	svctl modify nodeapps Command Parameters	A-65
A-55	svctl remove nodeapps Command Parameters	A-67

A-56	svctl setenv nodeapps Command Parameters	A-67
A-57	svctl start nodeapps Command Parameters	A-68
A-58	svctl stop nodeapps Command Parameters	A-69
A-59	svctl unsetenv nodeapps Command Parameters	A-69
A-60	svctl add ons Command Parameters	A-70
A-61	svctl modify ons Command Parameters	A-72
A-62	svctl add scan Command Parameters	A-73
A-63	svctl config scan Command Parameters	A-74
A-64	svctl modify scan Command Parameters	A-76
A-65	svctl relocate scan Command Parameters	A-77
A-66	svctl remove scan Command Parameters	A-77
A-67	svctl start scan Command Parameters	A-78
A-68	svctl add scan_listener Command Parameters	A-79
A-69	svctl config scan_listener Command Parameters	A-80
A-70	svctl modify scan_listener Command Parameters	A-82
A-71	svctl remove scan_listener Command Parameters	A-83
A-72	svctl relocate server Command Parameters	A-86
A-73	svctl add service Command Parameters	A-87
A-74	svctl config service Command Parameters	A-93
A-75	svctl disable service Command Parameters	A-94
A-76	svctl enable service Command Parameters	A-95
A-77	svctl modify service Parameters for Moving a Service	A-97
A-78	svctl modify service Parameters for Changing to a Preferred Instance	A-97
A-79	svctl modify service Parameters for Changing Status of Multiple Instances	A-98
A-80	svctl modify service Parameters	A-98
A-81	svctl predict service Command Parameters	A-102
A-82	svctl relocate service Command Parameters	A-103
A-83	svctl remove service Command Parameters	A-104
A-84	svctl status service Command Parameters	A-107
A-85	svctl stop service Command Parameters	A-108
A-86	svctl add srvpool Command Parameters	A-109
A-87	svctl modify srvpool Command Parameters	A-111
A-88	svctl remove srvpool Command Parameters	A-112
A-89	svctl add vip Command Parameters	A-113
A-90	svctl config vip Command Parameters	A-114
A-91	svctl getenv vip Command Parameters	A-115
A-92	svctl modify vip Command Parameters	A-116

A-93	svctl relocate vip Command Parameters	A-117
A-94	svctl remove vip Command Parameters	A-117
A-95	svctl setenv vip Command Parameters	A-118
A-96	svctl start vip Command Parameters	A-118
A-97	svctl status vip Command Parameters	A-119
A-98	svctl stop vip Command Parameters	A-120
A-99	svctl unsetenv vip Command Parameters	A-120
A-100	svctl config volume Command Parameters	A-121
A-101	svctl disable volume Command Parameters	A-122
A-102	svctl enable volume Command Parameters	A-123
A-103	svctl remove volume Command Parameters	A-124
A-104	svctl start volume Command Parameters	A-124
A-105	svctl status volume Command Parameters	A-125
A-106	svctl stop volume Command Parameters	A-126

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, including Oracle Real Application Clusters One Node (Oracle RAC One Node). 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 noncluster Oracle databases that appear in other Oracle documentation. Where necessary, this manual refers to platform-specific documentation. This Preface includes 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

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

Access to Oracle Support

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

Related Documents

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 RAC 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 RAC 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 postinstallation 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 Administrator's Reference for Linux and UNIX-Based Operating Systems*
- *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 12c 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.

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.
<i>italic</i>	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
monospace	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Changes in This Release for Oracle Real Application Clusters Administration and Deployment Guide

This preface includes:

- [Changes in Oracle Real Application Clusters Release 18c, Version 18.1](#)
- [Changes in Oracle Real Application Clusters 12c Release 2 \(12.2\)](#)
- [Changes in Oracle Real Application Clusters 12c Release 1 \(12.1\)](#)

Changes in Oracle Real Application Clusters Release 18c, Version 18.1

Following is a list of features that are new in the *Oracle Real Application Clusters Administration and Deployment Guide* for Oracle Real Application Clusters 18c (18.1).

Continuous Application Availability

Continuous application availability is achieved with a combination of planned maintenance and Transparent Application Continuity. Planned maintenance of a database includes draining and migration of database sessions before you begin any planned maintenance, without applications being aware of the activity. Transparent database session draining occurs when Application Continuity is enabled and fails sessions over to another database instance at points within the transaction that are safe, such as when an application submits a connection test or at request boundaries within a transaction.

Transparent Application Continuity transparently tracks and records session and transactional states so that the database session can be recovered following recoverable outages. This is done safely and with no reliance on application knowledge or code changes, enabling Application Continuity to be standard for your applications. Transparency is achieved by consuming a new state-tracking infrastructure that categorizes session state usage as the application issues user calls.

Oracle RAC Sharding

Oracle RAC Sharding affinitizes table partitions to Oracle RAC instances, and routes database requests that specify a partitioning key to the instance that logically holds the corresponding partition. This provides better cache utilization and dramatically reduces block pings across instances. The partitioning key can only be added to the most performance critical requests. Requests that don't specify the key still work transparently and can be routed to any instance. No changes to the database schema are required to enable this feature. Oracle RAC Sharding provides performance and scalability benefits with minimal application changes.

Scalable Sequences

Scalable sequences alleviate index leaf block contention when loading data into tables that use sequence values as keys.

 **See Also:**

Oracle Database Administrator's Guide

Changes in Oracle Real Application Clusters 12c Release 2 (12.2)

Following is a list of features that are new in the *Oracle Real Application Clusters Administration and Deployment Guide* for Oracle Real Application Clusters 12c release 2 (12.2).

SCAN Listener Supports HTTP Protocol

Currently, when a connection request comes in, the SCAN listener load balances among the handlers located on the same node. It then directly hands off the connection to the least loaded handler on the node. With this release, the SCAN listener is aware of the HTTP protocol so that it can redirect HTTP clients to the appropriate handler, which can reside on nodes in the cluster other than the node on which the SCAN listener resides.

IPv6 Support for Oracle Real Application Clusters on The Private Network

You can configure cluster nodes to use either IPv4- or IPv6-based IP addresses on a private network, and you can use more than one private network for a cluster.

 **See Also:**

Oracle Clusterware Administration and Deployment Guide

Extend Oracle Database QoS Management to Fully Support Administrator-Managed Databases

In this release, full Oracle Database Quality of Service Management (Oracle Database QoS Management) support is available by also supporting its Management mode. Oracle supports schema consolidation within an administrator-managed Oracle RAC database by adjusting the CPU shares of performance classes running in the database. Additionally, database consolidation is supported by adjusting CPU counts per databases hosted on the same physical servers.

As administrator-managed databases do not run in server pools, the ability to expand or shrink the number of instances by changing the server pool size that is supported in policy-managed database deployments is not available for administrator-managed databases. This deployment support is integrated into the Oracle Database QoS Management pages in Oracle Enterprise Manager Cloud Control.

 **See Also:**

Oracle Database Quality of Service Management User's Guide

Oracle Real Application Clusters Reader Nodes

Oracle RAC Reader Nodes facilitate Oracle Flex Cluster architecture by allocating a set of read/write instances running Online Transaction Processing (OLTP) workloads and a set of read-only database instances across Hub Nodes and Leaf Nodes in the cluster. In this architecture, updates to the read-write instances are immediately propagated to the read-only instances on the Leaf Nodes, where they can be used for online reporting or instantaneous queries.

 **See Also:**

Oracle Clusterware Administration and Deployment Guide

Server Weight-Based Node Eviction

Server weight-based node eviction acts as a tie-breaker mechanism in situations where Oracle Clusterware must evict a particular node or a group of nodes from a cluster, in which all nodes represent an equal choice for eviction. The server weight-based node eviction mechanism helps to identify the node or the group of nodes to be evicted based on additional information about the load on those servers. Two principle mechanisms, a system-inherent automatic mechanism and a user input-based mechanism, exist to provide respective guidance.

 **See Also:**

Oracle Clusterware Administration and Deployment Guide

Separation of Duty for Administering Oracle Real Application Clusters

Starting with Oracle Database 12c release 2 (12.2), Oracle Database provides support for separation of duty best practices when administering Oracle Real Application Clusters (Oracle RAC) by introducing the SYSRAC administrative privilege for the clusterware agent. This feature removes the need to use the powerful SYSDBA administrative privilege for Oracle RAC.

SYSRAC, like SYSDG, SYSBACKUP and SYSKM, helps enforce separation of duties and reduce reliance on the use of SYSDBA on production systems. This administrative privilege is the default mode for connecting to the database by the clusterware agent on behalf of the Oracle RAC utilities, such as SRVCTL.

In-Memory FastStart

In-Memory FastStart optimizes the population of database objects in the In-Memory column store by storing In-Memory compression units directly on disk.

**See Also:**

Oracle Database In-Memory Guide

Changes in Oracle Real Application Clusters 12c Release 1 (12.1)

The following are changes in *Oracle Real Application Clusters Administration and Deployment Guide* for Oracle Real Application Clusters (Oracle RAC) 12c:

- [Changes in Oracle Real Application Clusters 12c Release 1 \(12.1.0.2\)](#)
- [Changes in Oracle Real Application Clusters 12c Release 1 \(12.1.0.1\)](#)
- [Deprecated Features](#)
- [Desupported Features](#)

Changes in Oracle Real Application Clusters 12c Release 1 (12.1.0.2)

The following features are new in this release:

- **In-Memory Column Store**

The In-Memory Column Store is an optional area in the SGA that stores whole tables, table partitions, and individual columns in a compressed columnar format. The database uses special techniques, including SIMD vector processing, to scan columnar data extremely rapidly. The In-Memory Column Store is a supplement to, rather than a replacement for, the database buffer cache.

**See Also:**

Oracle Database Data Warehousing Guide for more information

- **In-Memory Transaction Manager**

The In-Memory Transaction Manager is an independent engine that automatically provides read consistency for transactions that apply changes to the In-Memory Column Store. This engine is necessary because tables and partitions residing in the In-Memory Column Store are stored in columnar format in memory and in row-major format in the data files and database buffer cache.

**See Also:**

Oracle Database Concepts for more information

- **Rapid Home Provisioning**

Rapid Home Provisioning enables you to deploy Oracle homes based on images stored in a catalog of precreated software homes.

 **See Also:**

Oracle Clusterware Administration and Deployment Guide for more information

- **Full Database In-Memory Caching**

In this release you can cache an entire database in memory. Use this feature when the buffer cache size of each instance is greater than the size of the whole database. In Oracle RAC systems, for well-partitioned applications, you can use this feature when the combined buffer caches of all database instances (with some extra space to handle duplicate cached blocks between instances) is greater than the database size.

 **See Also:**

Oracle Database Performance Tuning Guide for more information

- **Memory Guard Does Not Require Oracle Database QoS Management to be Active**

With this release, Memory Guard is enabled by default independent of whether you use Oracle Database Quality of Service Management (Oracle Database QoS Management). Memory Guard detects memory stress on a node and causes new sessions to be directed to other instances until the existing workload drains and frees memory. When free memory increases on the node, then services are enabled again to automatically accept new connections.

Changes in Oracle Real Application Clusters 12c Release 1 (12.1.0.1)

The following features are new in this release:

- **Application Continuity**

Before this release, application developers were required to deal explicitly with outages of the underlying software, hardware, and communications layers if they wanted to mask outages from end users.

In Oracle Database 10g, Fast Application Notification (FAN) quickly delivered exception conditions to applications. However, neither FAN nor earlier Oracle technology reported the outcome of the last transaction to the application, or recovered the in-progress request from an application perspective. As a result, outages were exposed leading to user inconvenience and lost revenue. Users could unintentionally make duplicate purchases and submit multiple payments for the same invoice. In the problematic cases, the administrator needed to reboot the mid-tier to deal with the incoming problems this caused.

Application Continuity is an application-independent feature that attempts to recover incomplete requests from an application perspective and masks many system, communication, hardware failures, and storage outages from the end user.

 **See Also:**

- *Oracle Database Concepts* for more information
- [Ensuring Application Continuity](#)

- **Transaction Guard for Java**

This feature exposes the new Application Continuity infrastructure to Java. It provides support for:

- At-most-once transaction execution protocol, such as transaction idempotence
- API for retrieving logical transaction ID (LTXID)
- Attribute to get Connection or Session status

 **See Also:**

Oracle Database JDBC Developer's Guide for more information

- **Transaction Idempotence**

This feature delivers a general purpose, application-independent infrastructure that enables recovery of work from an application perspective and masks most system, communication, and hardware failures from the user. Transaction idempotence ensures that your transactions are executed on time and, at most, once.

 **See Also:**

Oracle Database Development Guide for more information

- **Oracle Flex Clusters**

Large clusters consisting of, potentially, thousands of nodes, provide a platform for Oracle RAC to support massive parallel query operations.

 **See Also:**

Oracle Clusterware Administration and Deployment Guide for more information about Oracle Flex Clusters

- **Shared Oracle ASM Password File in a Disk Group**

This feature implements the infrastructure needed to address the bootstrapping issues of storing an Oracle Automatic Storage Management (Oracle ASM) shared password file in an Oracle ASM disk group.

 **See Also:**

Oracle Automatic Storage Management Administrator's Guide

- **Global Data Services**

Similar to the way Oracle RAC supports a database service and enables service-level workload management across database instances in a cluster, Global Data Services provides Oracle RAC-like connect-time and run-time load balancing, failover, and centralized service management for a set of replicated databases that offer common services. The set of databases can include Oracle RAC and nonclustered Oracle databases interconnected with Oracle Data Guard, Oracle GoldenGate, or any other replication technology.

 **See Also:**

Oracle Database Global Data Services Concepts and Administration Guide for more information

- **Shared Grid Naming Service**

One instance of Grid Naming Service (GNS) can service any number of clusters.

 **See Also:**

Oracle Clusterware Administration and Deployment Guide for more information

- **What-If Command Evaluation for Oracle RAC**

This feature of Oracle Clusterware improves resource management and availability, through a mechanism that provides a policy response to a hypothetical planned or unplanned event, without modifying the state of the system.

In Oracle RAC, enhancements to SRVCTL aid you in determining the impact of certain commands before you run them to determine the potential impact of the command.

 **See Also:**

- "[SRVCTL Usage Information](#)" for a list of SRVCTL commands with What-If functionality
- *Oracle Clusterware Administration and Deployment Guide* for a list of Oracle Clusterware Control (CRSCTL) utility commands with similar enhancements

- **Restricting Service Registration for Oracle RAC Deployments**

This feature allows listener registration only from local IPs by default and provides the ability to configure and dynamically update a set of IP addresses or subnets from which registration requests are allowed by the listener.

 **See Also:**

Oracle Database Net Services Administrator's Guide for more information

- **Restricting Service Registration with Valid Node Checking**

This feature enables the network administrator to specify a list of nodes and subnet information from which the Single Client Access Name (SCAN) listener accepts registration. You can specify the nodes and subnets information using SRVCTL, and SRVCTL stores the information in the SCAN listener resource profile and this information is also written to the `listener.ora` file. Restricting client access to a database makes Oracle RAC even more secure and less vulnerable to security threats and attacks.

- **Pluggable Databases**

Pluggable Databases enables an Oracle database to contain a portable collection of schemas, schema objects, and nonschema objects that appears to an Oracle Net client as a separate database. This self-contained collection is called a pluggable database (PDB). A container database (CDB) is an Oracle database that includes zero, one, or many user-created pluggable databases (PDBs). You can unplug a PDB from a CDB and plug it into a different CDB.

 **See Also:**

Oracle Database Administrator's Guide

- **Support of Oracle Home User on Windows**

Starting with Oracle Database 12c, Oracle Database supports the use of an Oracle home user, which you can specify at installation time. The Oracle home user is associated with a Windows domain user. The Windows domain user should be a low-privileged (non-Administrator) account to ensure that the Oracle home user has a limited set of privileges, thus ensuring that the Oracle Database services have only those privileges required to run Oracle products.

Windows Administrator user privileges are still required to perform Oracle software maintenance tasks including installation, upgrade, patching, and so on. Oracle Database administrative tools have been enhanced to ask for the password of the Oracle home user, if needed. In Oracle RAC environments, you can store the password for the Oracle home user in a secure wallet. If such a wallet exists, then the Oracle Database administrative tools automatically use the password from the wallet and do not require the user to enter the password for the Oracle home user.

 **See Also:**

Oracle Real Application Clusters Installation Guide for Microsoft Windows x64 (64-Bit)

- **Cluster Resources for Oracle ACFS and Oracle ADVM**

Oracle Clusterware resource support includes enhancements for Oracle homes stored on Oracle Automatic Storage Management Cluster File System (Oracle ACFS), Oracle ACFS General Purpose file systems for Grid homes, and Oracle ASM Dynamic Volume Manager (Oracle ADVM) volumes. These resources, that Oracle Clusterware manages, support automatic loading of Oracle ACFS, Oracle ADVM and OKS drivers, disk group mounts, dynamic volume enablement, and automatic Oracle ACFS file system mounts.

 **See Also:**

Oracle Automatic Storage Management Administrator's Guide for more information

- **Oracle Highly Available NFS**

You can configure Oracle ACFS as a highly available, exported file system service. This service uses Oracle ACFS' clusterwide data consistency and coherency, in combination with virtual IP addresses, to provide failover capability for NFS exports. By mounting the NFS export from this virtual IP address, a client can be assured that, if one node of the cluster is available, then the NFS export will be available.

 **See Also:**

Oracle Automatic Storage Management Administrator's Guide for more information

- **Policy-Based Cluster Management and Administration**

Oracle Grid Infrastructure allows running multiple applications in one cluster. Using a policy-based approach, the workload introduced by these applications can be allocated across the cluster using a policy. In addition, a policy set enables different policies to be applied to the cluster over time as required. You can define policy sets using a web-based interface or a command-line interface.

Hosting various workloads in the same cluster helps to consolidate the workloads into a shared infrastructure that provides high availability and scalability. Using a centralized policy-based approach allows for dynamic resource reallocation and prioritization as the demand changes.

 **See Also:**

Oracle Clusterware Administration and Deployment Guide for more information

- **Online Resource Attribute Modification**

Oracle Clusterware manages hardware and software components for high availability using a resource model. You use resource attributes to define how Oracle Clusterware manages those resources. You can modify certain resource

attributes and implement those changes without having to restart the resource using online resource attribute modification. You manage online resource attribute modification with certain SRVCTL and CRSCTL commands.

Deprecated Features

Deprecation of single-letter SRVCTL CLI options

All SRVCTL commands have been enhanced to accept full-word options instead of the single-letter options. All new SRVCTL command options added in this release support full-word options, only, and do not have single-letter equivalents. The use of single-letter options with SRVCTL commands might be desupported in a future release.

Desupported Features



See Also:

Oracle Database Upgrade Guide for more information

- **Oracle Cluster File System for Windows**
Oracle no longer supports Oracle Cluster File System (OCFS) on Windows.
- **Raw (block) storage devices for Oracle Database and related technologies**
Oracle Database 12c release 1 (12.1) and its related grid technologies, such as Oracle Clusterware, no longer support the direct use of raw or block storage devices. You must move existing files from raw or block devices to Oracle ASM before you upgrade to Oracle Clusterware 12c release 1 (12.1).

1

Introduction to Oracle RAC

Provides an overview of Oracle Real Application Clusters (Oracle RAC) installation and administration, and various components and functions.

This chapter includes the following topics:

- [Overview of Oracle RAC](#)
- [Overview of Installing Oracle RAC](#)
- [Overview of Oracle Real Application Clusters One Node](#)
- [Overview of Oracle Clusterware for Oracle RAC](#)
- [Overview of Oracle RAC Architecture and Processing](#)
- [Overview of Automatic Workload Management with Dynamic Database Services](#)
- [Overview of Server Pools and Policy-Managed Databases](#)
- [Overview of Oracle Database Quality of Service Management](#)
- [Overview of Hang Manager](#)
- [Overview of Oracle Multitenant with Oracle RAC](#)
- [Overview of Database In-Memory and Oracle RAC](#)
- [Overview of Managing Oracle RAC Environments](#)

Overview of Oracle RAC

Provides an introduction to Oracle RAC and its functionality.

Non-cluster 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 non-cluster 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.

A [cluster](#) comprises multiple interconnected computers or servers that appear as if they are one server to end users and applications. The Oracle RAC option with Oracle

¹ With Oracle Database 10g release 2 (10.2) and later releases, Oracle Clusterware supports 100 nodes in an Oracle Clusterware standard Cluster, with the option to run 100 database instances belonging to one production database on these nodes.

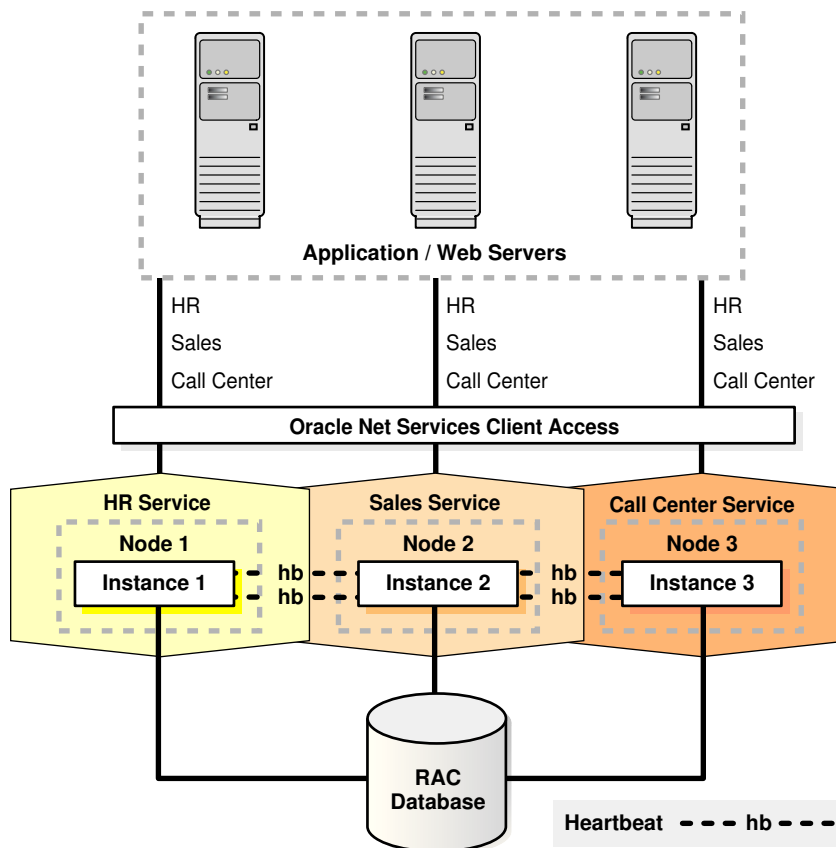
Database enables you to cluster Oracle databases. Oracle RAC uses Oracle Clusterware for the infrastructure to bind multiple servers so they operate as a single system.

Oracle Clusterware is a portable cluster management solution that is integrated with Oracle Database. Oracle Clusterware is a required component for using Oracle RAC that provides the infrastructure necessary to run Oracle RAC. Oracle Clusterware also manages resources, such as [Virtual Internet Protocol \(VIP\)](#) addresses, databases, listeners, services, and so on. In addition, Oracle Clusterware enables both noncluster Oracle databases and Oracle RAC databases to use the Oracle high-availability infrastructure. Oracle Clusterware along with Oracle Automatic Storage Management (Oracle ASM) (the two together comprise the [Oracle Grid Infrastructure](#)) enables you to create a clustered pool of storage to be used by any combination of noncluster and Oracle RAC databases.

Oracle Clusterware is the only clusterware that you need for most platforms on which Oracle RAC operates. If your database applications require vendor clusterware, then you can use such clusterware in conjunction with Oracle Clusterware if that vendor clusterware is certified for Oracle RAC.

[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 must run 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 [Oracle Extended 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 a single database cluster. In an extended cluster, the nodes in the cluster are typically dispersed, geographically, such as between two fire cells, between two rooms or buildings, or between two different data centers or cities. For availability reasons, the data must be located at both sites, thus requiring the implementation of 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 considering distance, latency, and the degree of protection it provides. Oracle RAC on extended clusters provides higher availability than is possible with local Oracle RAC configurations, but an extended 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 or 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 Data Guard Concepts and Administration* and on the Maximum Availability Architecture (MAA) Web site.

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 Installing Oracle RAC

Install Oracle Grid Infrastructure and Oracle Database software using Oracle Universal Installer, 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.

Alternatively, you can install Oracle RAC using Rapid Home Provisioning, which offers all of the advantages of Oracle Universal Installer and DBCA previously specified. In addition, Rapid Home Provisioning allows for standardization and automation.

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

- [Understanding Compatibility in Oracle RAC Environments](#)
- [Oracle RAC Database Management Styles and Database Installation](#)
- [Oracle RAC Database Management Styles and Database Creation](#)
- [Overview of Extending an Oracle RAC Cluster](#)

 **Note:**

You must first install Oracle Grid Infrastructure before installing Oracle RAC.

Understanding Compatibility in Oracle RAC Environments

To run Oracle RAC in configurations with different versions of Oracle Database in the same cluster, you must first install Oracle Grid Infrastructure, which must be the same version, or higher, as the highest version of Oracle Database that you want to deploy in this cluster. For example, to run an Oracle RAC 12c database and an Oracle RAC 18c database in the same cluster, you must install Oracle Grid Infrastructure 18c. Contact My Oracle Support for more information about version compatibility in Oracle RAC environments.

 **Note:**

Oracle does not support deploying an Oracle9i cluster in an Oracle Grid Infrastructure 12c, or later, environment.

Oracle RAC Database Management Styles and Database Installation

Before installing the Oracle RAC database software and creating respective databases, decide on the management style you want to apply to the Oracle RAC databases, as described in "Overview of Server Pools and Policy-Managed Databases".

The management style you choose impacts the software deployment and database creation. If you choose the administrator-managed database deployment model, using a per-node installation of software, then it is sufficient to deploy the Oracle Database software (the database home) on only those nodes on which you plan to run Oracle Database.

If you choose the policy-managed deployment model, using a per-node installation of software, then you must deploy the software on all nodes in the cluster, because the dynamic allocation of servers to server pools, in principle, does not predict on which server a database instance can potentially run. To avoid instance startup failures on servers that do not host the respective database home, Oracle strongly recommends that you deploy the database software on all nodes in the cluster. When you use a shared Oracle Database home, accessibility to this home from all nodes in the cluster is assumed and the setup needs to ensure that the respective file system is mounted on all servers, as required.

Oracle Universal Installer will only allow you to deploy an Oracle Database home across nodes in the cluster if you previously installed and configured Oracle Grid Infrastructure for the cluster. If Oracle Universal Installer does not give you an option to deploy the database home across all nodes in the cluster, then check the prerequisite, as stated, by Oracle Universal Installer.

During installation, you can choose to create a database during the database home installation. Oracle Universal Installer runs DBCA to create your Oracle RAC database according to the options that you select.

 **See Also:**

"Oracle RAC Database Management Styles and Database Creation" for more information if you choose this option

 **Note:**

Before you create a database, a default listener must be running in the Oracle Grid Infrastructure home. If a default listener is not present in the Oracle Grid Infrastructure home, then DBCA returns an error instructing you to run NETCA from the Oracle Grid Infrastructure home to create a default listener.

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

Oracle RAC Database Management Styles and Database Creation

Part of Oracle Database deployment is the creation of the database.

You can choose to create a database as part of the database software deployment, as described in "Oracle RAC Database Management Styles and Database Installation", or you can choose to only deploy the database software, first, and then, subsequently, create any database that is meant to run out of the newly created Oracle home by using DBCA. In either case, you must consider the management style that you plan to use for the Oracle RAC databases.

For administrator-managed databases, you must ensure that the database software is deployed on the nodes on which you plan to run the respective database instances. You must also ensure that these nodes have access to the storage in which you want to store the database files. Oracle recommends that you select Oracle ASM during database installation to simplify storage management. Oracle ASM automatically manages the storage of all database files within disk groups. If you plan to use Oracle Database Standard Edition to create an Oracle RAC database, then you *must* use Oracle ASM to store all of the database files.

For policy-managed databases, you must ensure that the database software is deployed on all nodes on which database instances can potentially run, given your active server pool setup. You must also ensure that these nodes have access to the storage in which you want to store the database files. Oracle recommends using Oracle ASM, as previously described for administrator-managed databases.

Server pools are a feature of Oracle Grid Infrastructure (specifically Oracle Clusterware). There are different ways you can set up server pools on the Oracle Clusterware level, and Oracle recommends you create server pools for database management before you create the respective databases. DBCA, however, will present you with a choice of either using precreated server pools or creating a new server pool, when you are creating a policy-managed database. Whether you can create a new server pool during database creation depends on the server pool configuration that is active at the time.

By default, DBCA creates one service for your Oracle RAC installation. This is the default database service and should not be used for user connectivity. 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:**

Oracle recommends that you reserve the default database service for maintenance operations and create dynamic database services for user or application connectivity as a post-database-creation step, using either SRVCTL or Oracle Enterprise Manager. DBCA no longer offers a dynamic database service creation option for Oracle RAC databases. For Oracle RAC One Node databases, you must create at least one dynamic database service.

Overview of Extending an Oracle RAC Cluster

If you want to extend the Oracle RAC cluster (also known as cloning) and add nodes to the existing environment after your initial deployment, then you must do this on multiple layers, considering the management style that you currently use in the cluster.

Oracle provides various means of extending an Oracle RAC cluster. In principle, you can choose from the following approaches to extend the current environment:

- Rapid Home Provisioning to provision new Oracle RAC databases and other software
- Cloning using cloning scripts
- Adding nodes using the `addnode.sh` (`addnode.bat` on Windows) script

Both approaches are applicable, regardless of how you initially deployed the environment. Both approaches copy the required Oracle software on to the node that you plan to add to the cluster. Software that gets copied to the node includes the Oracle Grid Infrastructure software and the Oracle database homes.

For Oracle database homes, you must consider the management style deployed in the cluster. For administrator-managed databases, you must ensure that the database software is deployed on the nodes on which you plan to run the respective database instances. For policy-managed databases, you must ensure that the database software is deployed on all nodes on which database instances can potentially run, given your active server pool setup. In either case, you must first deploy Oracle Grid Infrastructure on all nodes that are meant to be part of the cluster.

 **Note:**

Oracle cloning is not a replacement for cloning using Oracle Enterprise Manager as part of the Provisioning Pack. When you clone Oracle RAC using Oracle Enterprise Manager, the provisioning process includes a series of steps where details about the home you want to capture, the location to which you want to deploy, and various other parameters are collected.

For new installations or if you install only one Oracle RAC database, use the traditional automated and interactive installation methods, such as Oracle Universal Installer, Rapid Home Provisioning, 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 procedures detailed in "Adding and Deleting Oracle RAC from Nodes on Linux and UNIX Systems".

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.

Overview of Oracle Real Application Clusters One Node

Oracle Real Application Clusters One Node (Oracle RAC One Node) is an option to Oracle Database Enterprise Edition available since Oracle Database 11g release 2 (11.2).

Oracle RAC One Node is a single instance of an Oracle RAC-enabled database running on one node in the cluster, only, under normal operations. This option adds to the flexibility that Oracle offers for database consolidation while reducing management overhead by providing a standard deployment for Oracle databases in the enterprise. Oracle RAC One Node database requires Oracle Grid Infrastructure and, therefore, requires the same hardware setup as an Oracle RAC database.

Oracle supports Oracle RAC One Node on all platforms on which Oracle RAC is certified. Similar to Oracle RAC, Oracle RAC One Node is certified on Oracle Virtual Machine (Oracle VM). Using Oracle RAC or Oracle RAC One Node with Oracle VM increases the benefits of Oracle VM with the high availability and scalability of Oracle RAC.

With Oracle RAC One Node, there is no limit to server scalability and, if applications grow to require more resources than a single node can supply, then you can upgrade your applications online to Oracle RAC. If the node that is running Oracle RAC One Node becomes overloaded, then you can relocate the instance to another node in the cluster. With Oracle RAC One Node you can use the Online Database Relocation feature to relocate the database instance with no downtime for application users. Alternatively, you can limit the CPU consumption of individual database instances per server within the cluster using Resource Manager Instance Caging and dynamically change this limit, if necessary, depending on the demand scenario.

Using the Single Client Access Name (SCAN) to connect to the database, clients can locate the service independently of the node on which it is running. Relocating an Oracle RAC One Node instance is therefore mostly transparent to the client, depending on the client connection. Oracle recommends to use either Application

Continuity and Oracle Fast Application Notification or Transparent Application Failover to minimize the impact of a relocation on the client.

Oracle RAC One Node databases are administered slightly differently from Oracle RAC or non-cluster databases. For administrator-managed Oracle RAC One Node databases, you must monitor the candidate node list and make sure a server is always available for failover, if possible. Candidate servers reside in the Generic server pool and the database and its services will fail over to one of those servers.

For policy-managed Oracle RAC One Node databases, you must ensure that the server pools are configured such that a server will be available for the database to fail over to in case its current node becomes unavailable. In this case, the destination node for online database relocation must be located in the server pool in which the database is located. Alternatively, you can use a server pool of size 1 (one server in the server pool), setting the minimum size to 1 and the importance high enough in relation to all other server pools used in the cluster, to ensure that, upon failure of the one server used in the server pool, a new server from another server pool or the Free server pool is relocated into the server pool, as required.

 **Note:**

- Oracle RAC One Node supports Transaction Guard and Application Continuity for failing clients over.
- To prepare for all failure possibilities, you must add at least one Dynamic Database Service (Oracle Clusterware-managed database service) to an Oracle RAC One Node database.

Overview of Oracle Clusterware for Oracle RAC

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 advanced capabilities. Oracle Database also continues to support select third-party clusterware products on specified platforms.

Oracle Clusterware is designed for, and tightly integrated with, Oracle RAC. You can use Oracle Clusterware to manage high-availability operations in a cluster. 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 required components such as the VIP address, the Single Client Access Name (SCAN) (which includes the SCAN VIPs and the SCAN listener), Oracle Notification Service, and the Oracle Net listeners. These resources are automatically started when the node starts and automatically restart if they fail. The Oracle Clusterware daemons run on each node.

Anything that Oracle Clusterware manages is known as a *CRS resource*. A CRS resource can be a database, an instance, a service, a listener, a VIP address, or an application process. 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 any Oracle-defined CRS resources. Oracle Clusterware provides the framework that enables you to create CRS resources to manage any process running on servers in the cluster which are not predefined by Oracle. Oracle Clusterware stores the information that describes the configuration of these components in OCR that you can administer.

Another advantage with Oracle Flex Clusters is that you can convert Hub Nodes to Leaf Nodes and vice versa, and you can convert a Leaf Node to a Hub Node with no downtime.

This section includes the following topics:

- [Overview of Oracle Flex Clusters](#)
- [Overview of Reader Nodes](#)
- [Overview of Local Temporary Tablespaces](#)

Overview of Oracle Flex Clusters

Oracle Flex Clusters provide a platform for a variety of applications, including Oracle RAC databases with large numbers of nodes.

Oracle Flex Clusters also provide a platform for other service deployments that require coordination and automation for high availability.

All nodes in an Oracle Flex Cluster belong to a single Oracle Grid Infrastructure cluster. This architecture centralizes policy decisions for deployment of resources based on application needs, to account for various service levels, loads, failure responses, and recovery.

Oracle Flex Clusters contain two types of nodes arranged in a hub and spoke architecture: [Hub Nodes](#) and [Leaf Nodes](#). The number of Hub Nodes in an Oracle Flex Cluster can be as many as 64. The number of Leaf Nodes can be many more. Hub Nodes and Leaf Nodes can host different types of applications.

Overview of Reader Nodes

Reader nodes are instances of an Oracle RAC database that run on Leaf Nodes in an Oracle Flex Cluster.

An advantage of using reader nodes is that these instances are not affected if you need to reconfigure a Hub Node. Parallel query jobs running on reader nodes are not subject to the brownout times that can occur when you reconfigure a Hub Node, and can continue to service clients that are connected to the reader nodes, as long as the Hub Node to which it is connected is not evicted from the cluster. You can scale up to 64 reader nodes per Hub Node, and reader nodes can utilize massive parallel query to speed up operations on large data sets.

Database instances running on reader nodes have different characteristics than database instances running on Hub Nodes. Hub Node instances are similar to previous versions of Oracle RAC database instances, whereas instances running on reader nodes have the following notable differences:

- Database instances run in read-only mode.
- Database instances are unaffected if you reconfigure Hub Node clusterware. Database instances running on reader nodes are not subject to the brownout times and can continue to service the clients connected to the reader nodes, as long as the Hub Node to which it is connected is not evicted from the cluster. Using reader nodes, you can scale up to 64 reader nodes per hub.
- Because database instances running on the reader nodes are read only, you will receive an error if you attempt any DDL or DML operations.

You can create services to direct queries to read-only instances running on reader nodes. These services can use parallel query to further speed up performance. Oracle recommends that you size the memory in these reader nodes as high as possible so that parallel queries can use the memory for best performance.

Overview of Local Temporary Tablespaces

Oracle uses local temporary tablespaces to write spill-overs to the local (non-shared) temporary tablespaces which are created on local disks on the reader nodes.

It is still possible for SQL operations, such as hash aggregation, sort, hash join, creation of cursor-duration temporary tables for the `WITH` clause, and star transformation to spill over to disk (specifically to the global temporary tablespace on shared disks). Management of the local temporary tablespace is similar to that of the existing temporary tablespace.

Local temporary tablespaces improve temporary tablespace management in read-only instances by:

- Storing temp files in reader node private storage to take advantage of the I/O benefits of local storage.
- Avoiding expensive cross-instance temporary tablespace management.
- Increased addressability of temporary tablespace.
- Improving instance warm-up performance by eliminating on-disk space metadata management.



Note:

You cannot use local temporary tablespaces to store database objects, such as tables or indexes. This same restriction is also true for space for Oracle global temporary tables.

This section includes the following topics:

- [Parallel Execution Support for Cursor-Duration Temporary Tablespaces](#)
- [Local Temporary Tablespace Organization](#)
- [Temporary Tablespace Hierarchy](#)
- [Local Temporary Tablespace Features](#)
- [Metadata Management of Local Temporary Files](#)
- [DDL Support for Local Temporary Tablespaces](#)

- [Local Temporary Tablespaces for Users](#)
- [Atomicity Requirement for Commands](#)
- [Local Temporary Tablespace and Dictionary Views](#)

Parallel Execution Support for Cursor-Duration Temporary Tablespaces

The temporary tablespaces created for the `WITH` clause and star transformation exist in the temporary tablespace on the shared disk. A set of parallel query child processes load intermediate query results into these temporary tablespaces, which are then read later by a different set of child processes. There is no restriction on how these child processes reading these results are allocated, as any parallel query child process on any instance can read the temporary tablespaces residing on the shared disk.

For read-write and read-only instance architecture, as the parallel query child processes load intermediate results to the local temporary tablespaces of these instances, the parallel query child processes belonging to the instance where the intermediate results are stored share affinity with the reads for the intermediate results and can thus read them.

Local Temporary Tablespace Organization

You can create local temporary tablespace using the `{FOR_RIM | FOR_ALL}` extension, as follows:

```
CREATE LOCAL TEMPORARY TABLESPACE FOR RIM local_temp_ts_for_rim TEMPFILE\  
  '/u01/app/oracle/database/12.2.0.1/dbs/temp_file_rim'\  
  EXTENT MANAGEMENT LOCAL UNIFORM SIZE 1M AUTOEXTEND ON;
```

- Creation of a local temporary tablespace results in the creation of local temporary files on every instance and not a single file, as is currently true for shared global temporary tablespaces.
- You can create local temporary tablespaces for both read-only and read-write instances. The `LOCAL...FOR_RIM` option creates a local temporary tablespace whose files reside on local disks for read-only instances. The `LOCAL...FOR_ALL` option creates a local temporary tablespace whose files reside on local disks for both read-write and read-only instances. For example:

```
CREATE LOCAL TEMPORARY TABLESPACE FOR ALL temp_ts_for_all TEMPFILE\  
  '/u01/app/oracle/database/12.2.0.1/dbs/temp_file_all'\  
  EXTENT MANAGEMENT LOCAL UNIFORM SIZE 1M AUTOEXTEND ON;
```

Temporary Tablespace Hierarchy

When you define local temporary tablespace and shared (existing) temporary tablespace, there is a hierarchy in which they are used. To understand the hierarchy, remember that there can be multiple shared temporary tablespaces in a database, such the default shared temporary tablespace for the database and multiple temporary tablespaces assigned to individual users. If a user has a shared temporary tablespace assigned, then that tablespace is used first, otherwise the database default temporary tablespace is used.

Once a tablespace has been selected for spilling during query processing, there is no switching to another tablespace. For example, if a user has a shared temporary tablespace assigned and during spilling it runs out of space, then there is no switching to an alternative tablespace. The spilling, in that case, will result in an error. Additionally, remember that shared temporary tablespaces are shared among instances.

The allocation of temporary space for spilling to a local temporary tablespace differs between read-only and read-write instances. For read-only instances, the following is the priority of selecting which temporary location to use for spills:

1. Allocate from a user's local temporary tablespace.
2. Allocate from the database default local temporary tablespace.
3. Allocate from a user's temporary tablespace.
4. Allocate from the database default temporary tablespace.

 **Note:**

If there is no local temporary tablespace in the database, then read-only instances will spill to shared temporary tablespace.

For read-write instances, the priority of allocation differs from the preceding allocation order, as shared temporary tablespaces are given priority, as follows:

1. Allocate from a user's shared temporary tablespace.
2. Allocate from a user's local temporary tablespace.
3. Allocate from the database default shared temporary tablespace.
4. Allocate from the database default local temporary tablespace.

 **Note:**

For read-write instances to use a local temporary tablespace, that temporary tablespace must be created with the `LOCAL...FOR ALL` option.

Local Temporary Tablespace Features

Instances cannot share local temporary tablespace, hence one instance cannot take local temporary tablespace from another. If an instance runs out of temporary tablespace during spilling, then the statement results in an error.

- Local temporary tablespace support only one `BIGFILE` per tablespace.
- To address contention issues arising from having only one `BIGFILE`-based local temporary tablespace, multiple local temporary tablespaces can be assigned to different users, as default.
- A database administrator can specify the default temporary tablespace for a user using `ALTER USER` syntax. For example:

```
ALTER USER MAYNARD LOCAL TEMPORARY TABLESPACE temp_ts_for_rim;
```

- A user can be configured with two default temporary tablespaces:
 - One local temporary (created with the `FOR RIM` option) when the user is connected to the read-only instance running on reader nodes.
 - One shared temporary tablespace to be used when the same user is connected on the read-write instances running on a Hub Node.

Metadata Management of Local Temporary Files

Currently, temporary file information (such as file name, creation size, creation SCN, temporary block size, and file status) is stored in the control file along with the initial and max files, as well as auto extent attributes. However, the information about local temporary files in the control file is common to all applicable instances (read write and read only for LOCAL...FOR ALL, and read only for LOCAL...FOR RIM).

Instance-specific information, such as bitmap for allocation, current size for a temporary file, and the file status, is stored in the SGA on instances and not in the control file because this information can be different for different instances. When an instance starts up, it reads the information in the control file and creates the temporary files that constitute the local temporary tablespace for that instance. If there are two or more instances running on a node, then each instance will have its own local temporary files.

For local temporary tablespaces, there is a separate file for each involved instance. If you create the local temporary tablespace using the LOCAL...FOR RIM option, then Oracle creates files only on read-only instances. If you create the local temporary tablespace using the LOCAL...FOR ALL option, then Oracle creates files on all instances. The local temporary file names follow a naming convention such that the instance numbers are appended to the temporary file names specified while creating the local temporary tablespace.

For example, assume that a read-only node, N1, runs two Oracle read-only database instances with numbers 3 and 4. The following DDL command creates two files on node N1—/temp/temp_file_rim_3 and /temp/temp_file_rim_4, for instance 3 and 4 respectively:

```
CREATE LOCAL TEMPORARY TABLESPACE FOR RIM temp_ts_for_rim TEMPFILE '/temp/
temp_file_rim'\
    EXTENT MANAGEMENT LOCAL UNIFORM SIZE 1M AUTOEXTEND ON;
```

Assuming that there are two read-write instances (instance numbers 1 and 2) and two read-only instances (instance numbers 3 and 4), the following DDL command creates four files—/temp/temp_file_all_1 and /temp/temp_file_all_2 for instances 1 and 2, respectively, and /temp/temp_file_all_3 and /temp/temp_file_all_4 for instances 3 and 4, respectively:

```
CREATE LOCAL TEMPORARY TABLESPACE FOR ALL temp_ts_for_all TEMPFILE '/temp/
temp_file_all'\
    EXTENT MANAGEMENT LOCAL UNIFORM SIZE 1M AUTOEXTEND ON;
```

DDL Support for Local Temporary Tablespaces

You manage local temporary tablespaces and temporary files with either ALTER TABLESPACE, ALTER DATABASE, or both DDL commands. All DDL commands related to local temporary tablespace management and creation are run from the read-write instances. Running all other DDL commands will affect all instances in a homogeneous manner.

For example, the following command resizes the temporary files on all read-only instances:

```
ALTER TABLESPACE temp_ts_for_rim RESIZE 1G;
```

For local temporary tablespaces, Oracle supports the allocation options and their restrictions currently active for temporary files.

To run a DDL command on a local temporary tablespace on a read-only instance, only (in particular, `CREATE LOCAL TABLESPACE...FOR RIM` or `ALTER TABLESPACE LOCAL TABLESPACE...FOR RIM`), there must be at least one read-only instance in the cluster. This restriction is not applicable when creating or altering local temporary tablespaces `FOR ALL`. Users can assign a default local temporary tablespace to the database with a `DEFAULT LOCAL TEMPORARY TABLESPACE` clause appended to the `ALTER DATABASE` command.

**Note:**

The annotation `FOR RIM` is unnecessary because it is implied by the name of the table space.

For example:

```
ALTER DATABASE DEFAULT LOCAL TEMPORARY TABLESPACE temp_ts_for_rim;
```

A database administrator can specify default temporary tablespace when creating the database, as follows:

```
CREATE DATABASE .. DEFAULT TEMPORARY TABLESPACE temp_ts_for_dbtemp_ts_for_rim
TEMPFILE\
  '/temp/temp_file_for_dbrim' EXTENT MANAGEMENT LOCAL UNIFORM SIZE 1M AUTOEXTEND ON;
```

It is not possible to specify default local temporary tablespaces using the `CREATE DATABASE` command. When you create a database, its default local temporary tablespace will point to the default shared temporary tablespace. Database administrators must run the `ALTER DATABASE` command to assign an existing local temporary tablespace as the default for the database.

Local Temporary Tablespace for Users

When you create a user without explicitly specifying shared or local temporary tablespace, the user inherits shared and local temporary tablespace from the corresponding default database tablespaces. You can specify default local temporary tablespace for a user, as follows:

```
CREATE USER new_user IDENTIFIED BY new_user LOCAL TEMPORARY TABLESPACE
temp_ts_for_all;
```

You can change the local temporary tablespace for a user using the `ALTER USER` command, as follows:

```
ALTER USER maynard LOCAL TEMPORARY TABLESPACE temp_ts_for_rim;
```

As previously mentioned, default user local temporary tablespace can be shared temporary space. Consider the following items for the `ALTER USER...TEMPORARY TABLESPACE` command:

- You can change the user default local temporary tablespace to any existing local temporary tablespace.
- If you want to set the user default local temporary tablespace to a shared temporary tablespace, `T`, then `T` must be the same as the default shared temporary tablespace.
- If a default user local temporary tablespace points to a shared temporary tablespace, then, when you change the default shared temporary tablespace of

the user, you also change the default local temporary tablespace to that tablespace.

Following are some examples of local temporary space management using the `ALTER` command:

- To take a local temporary tablespace offline:

```
ALTER DATABASE TEMPFILE '/temp/temp_file_rim' OFFLINE;
```

- To decrease the size of a local temporary tablespace:

```
ALTER TABLESPACE temp_ts_for_rim SHRINK SPACE KEEP 20M
```

- To change the auto-extension attributes of a local temporary file:

```
ALTER TABLESPACE temp_ts_for_rim AUTOEXTEND ON NEXT 20G
```

- To resize a local temporary file:

```
ALTER TABLESPACE temp_ts_for_rim RESIZE 10G
```

 **Note:**

When you resize a local temporary file, it applies to individual files.

Some read-only instances may be down when you run any of the preceding commands. This does not prevent the commands from succeeding because, when a read-only instance starts up later, it creates the temporary files based on information in the control file. Creation is fast because Oracle reformats only the header block of the temporary file, recording information about the file size, among other things. If you cannot create any of the temporary files, then the read-only instance stays down. Commands that were submitted from a read-write instance are replayed, immediately, on all open read-only instances.

Atomicity Requirement for Commands

All the commands that you run from the read-write instances are performed in an atomic manner, which means the command succeeds only when it succeeds on all live instances.

Local Temporary Tablespace and Dictionary Views

Oracle extended dictionary views to display information about local temporary tablespaces. Oracle made the following changes:

- All the diagnosability information related to temporary tablespaces and temporary files exposed through AWR, SQL monitor, and other utilities, is also available for local temporary tablespaces and local temporary files. This information is available with the existing dictionary views for temporary tablespaces and temporary files: `DBA_TEMP_FILES`, `DBA_TEMP_FREE_SPACE`.
- The `USER_TABLESPACES` and `DBA_TABLESPACES` dictionary view are extended by a column, called `SHARED`, that indicates whether the temporary file is local (`FOR RIM` or `FOR ALL`) or shared.
- The `DBA_TEMP_FILES` dictionary view is extended by two columns: `SHARED` and `INST_ID`. The `SHARED` column indicates whether the temp file is local (`FOR RIM` or `FOR ALL`) or shared. The `INST_ID` column contains the instance number. For shared temporary files, there is a single row per file, and the `INST_ID` is null. For local

temporary files, this column contains information about temporary files per instance, such as the size of the file in bytes (`BYTES` column).

 **Note:**

`LOCAL...FOR RIM` shows only read-only instances per file.

- The `DBA_TEMP_FREE_SPACE` dictionary view is extended by two columns: `SHARED` and `INST_ID`. The `SHARED` column indicates whether the temporary file is local (`FOR RIM` or `FOR ALL`) or shared. The `INST_ID` column contains the instance number. For shared temporary files, there is a single row per file, and the `INST_ID` is null. For local temporary files, this column contains information about temporary files per instance, such as total free space available (`FREE_SPACE` column).

 **Note:**

`LOCAL...FOR RIM` shows only read-only instances per file.

- In the dictionary views, such as `DBA_TABLESPACES`, Oracle distinguishes the type of the tablespace using the `SHARED` column with the following values:
 - `SHARED`: for shared temporary tablespace
 - `LOCAL_ON_ALL`: for local temporary tablespace on all instances
 - `LOCAL_ON_RIM`: for local temporary tablespace on read-only instances

 **Note:**

Currently, spills onto temporary tablespace for queries (such as sort and hash join spills) are automatically encrypted. This is also true for spills to local temporary tablespace.

Overview of Oracle RAC Architecture and Processing

At a minimum, Oracle RAC requires Oracle Clusterware software infrastructure to 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](#)
- [Oracle RAC and Network Connectivity](#)
- [Overview of Using Dynamic Database Services to Connect to Oracle Databases](#)
- [Restricted Service Registration in Oracle RAC](#)
- [About Oracle RAC Software Components](#)

- [About Oracle RAC Background Processes](#)

Understanding Cluster-Aware Storage Solutions

An Oracle RAC database is a [shared everything](#) database. All data files, control files, SPFILES, 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. 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 is certified for use on a variety of storage architectures. It is your choice how to configure your storage, but you must use a supported cluster-aware storage solution. Oracle Database provides the following storage options for Oracle RAC:

- Oracle Automatic Storage Management (Oracle ASM)
Oracle recommends this solution to manage your storage.
- A certified cluster file system
 - Oracle recommends Oracle Automatic Storage Management Cluster File System (Oracle ACFS).
 - A third-party cluster file system on a cluster-aware volume manager that is certified for Oracle RAC. For example:
 - * Oracle OCFS2 (Linux, only)
 - * IBM GPFS (IBM AIX, only)
- Certified network file system (NFS) solution

Oracle RAC and Network Connectivity

All nodes in an Oracle RAC environment must connect to at least one Local Area Network (LAN) (commonly referred to as the *public network*) to enable users and applications to access the database.

In addition to the public network, Oracle RAC requires private network connectivity used exclusively for communication between the [nodes](#) and database instances running on those nodes. This network is commonly referred to as the *interconnect*.

The interconnect network is a private network that connects all of the servers in the cluster. The interconnect network must use at least one switch and a Gigabit Ethernet adapter.

Note:

- Oracle supports interfaces with higher bandwidth but *does not* support using crossover cables with the interconnect.
- Do not use the interconnect (the private network) for user communication, because [Cache Fusion](#) uses the interconnect for interinstance communication.

You can configure Oracle RAC to use either the User Datagram Protocol (UDP) or Reliable Data Socket (RDS) protocols for inter-instance communication on the interconnect. Oracle Clusterware uses the same interconnect using the UDP protocol, but cannot be configured to use RDS.

An additional network connectivity is required when using [Network Attached Storage \(NAS\)](#). Network attached storage can be typical NAS devices, such as NFS filers, or can be storage that is connected using Fibre Channel over IP, for example. This additional network communication channel should be independent of the other communication channels used by Oracle RAC (the public and private network communication). If the storage network communication must be converged with one of the other network communication channels, then you must ensure that storage-related communication gets first priority.

Overview of Using Dynamic Database Services to Connect to Oracle Databases

Applications should use the Dynamic Database Services feature to connect to an Oracle database over the public network.

Dynamic 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.

Users can access an Oracle RAC database using a client/server configuration or through one or more middle tiers, with or without connection pooling. By default, a user connection to an Oracle RAC database is established using the TCP/IP protocol but Oracle supports other protocols. Oracle RAC database instances must be accessed through the SCAN for the cluster.

Overview of Virtual IP Addresses

Oracle Clusterware hosts node virtual IP (VIP) addresses on the public network. Node VIPs are VIP addresses that clients use to connect to an Oracle RAC database. A typical connect attempt from a database client to an Oracle RAC database instance can be summarized, as follows:

1. The database client connects to SCAN (which includes a SCAN VIP on a public network), providing the SCAN listener with a valid service name.
2. The SCAN listener then determines which database instance hosts this service and routes the client to the local or node listener on the respective node.
3. The node listener, listening on a node VIP and a given port, retrieves the connection request and connects the client to the an instance on the local node.

If multiple public networks are used on the cluster to support client connectivity through multiple subnets, then the preceding operation is performed within a given subnet.

If a node fails, then the VIP address fails over to another node on which the VIP address can accept TCP connections, but it does not accept connections to the Oracle database. Clients that attempt to connect to a VIP address not residing on its home node receive a *rapid connection refused* error instead of waiting for TCP connect timeout messages. When the network on which the VIP is configured comes back

online, Oracle Clusterware fails back the VIP to its home node, where connections are accepted. 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

Oracle RAC 12c supports multiple public networks to enable access to the cluster through different subnets. Each network resource represents its own subnet and each database service uses a particular network to access the Oracle RAC database. Each network resource is a resource managed by Oracle Clusterware, which enables the VIP behavior previously described.

SCAN is a single network name defined either in your organization's Domain Name Server (DNS) or in the Grid Naming Service (GNS) that round robins to three IP addresses. Oracle recommends that all connections to the Oracle RAC database use the SCAN in their client connection string. Incoming connections are load balanced across the active instances providing the requested service through the three SCAN listeners. With SCAN, you do not have to change the client connection even if the configuration of the cluster changes (nodes added or removed). Unlike in previous releases, SCAN in Oracle RAC 12c fully supports multiple subnets, which means you can create one SCAN for each subnet in which you want the cluster to operate.

Restricted Service Registration in Oracle RAC

The valid node checking feature provides the ability to configure and dynamically update a set of IP addresses or subnets from which registration requests are allowed by the listener.

Database instance registration with a listener succeeds only when the request originates from a valid node. The network administrator can specify a list of valid nodes, excluded nodes, or disable valid node checking. The list of valid nodes explicitly lists the nodes and subnets that can register with the database. The list of excluded nodes explicitly lists the nodes that cannot register with the database. The control of dynamic registration results in increased manageability and security of Oracle RAC deployments.

By default, valid node checking for registration (VNCR) is enabled. In the default configuration, registration requests from all nodes within the subnet of the SCAN listener can register with the listener. Non-SCAN listeners only accept registration from instances on the local node. Remote nodes or nodes outside the subnet of the SCAN listener must be included on the list of valid nodes by using the `registration_invited_nodes_alias` parameter in the `listener.ora` file or by modifying the SCAN listener using `SRVCTL`.

About Oracle RAC Software Components

Oracle RAC databases generally 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 noncluster Oracle database and additional processes 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 In-Memory Transaction Manager integrates with the Cache Fusion protocol.
- The SGA size requirements for Oracle RAC are greater than the SGA requirements for noncluster Oracle databases due to Cache Fusion.

To ensure that each Oracle RAC database instance obtains the block that it requires 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 manage access to Cache Fusion resources and enqueue recovery processing.\

Cache Fusion monitors the latency on the private networks and the service time on the disks, and automatically chooses the best path. If shared disks include low latency SSDs, then Oracle automatically chooses the best path.

About Oracle RAC 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 `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 if a failure occurs.
- **GTX0-j: Global Transaction Process**
The `GTX0-j` process provides transparent support for XA global transactions in an Oracle 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 data file 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 `RMSn` processes perform manageability tasks for Oracle RAC. Tasks accomplished by an `RMSn` process include creation of resources related to 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 Database components that this section describes are in addition to the components that are described for noncluster Oracle databases in *Oracle Database Concepts*.

Overview of Automatic Workload Management with Dynamic Database Services

Services represent groups of applications with common attributes, service level thresholds, and priorities.

Application functions can be divided into workloads identified by services. For example, Oracle E-Business Suite can define a service for each responsibility, such as general ledger, accounts receivable, order entry, and so on. A service can span one or more instances of an Oracle database, multiple databases in a global cluster, and a single instance can support multiple services. The number of instances that are serving the service is transparent to the application. Services provide a single system image to manage competing applications, and allow each workload to be managed as a unit.

Middle tier applications and clients select a service by specifying the service name as part of the connection in the TNS connect string. For example, data sources for Oracle WebLogic Server are set to route to a service. Using Net Easy*Connection, this connection comprises simply the service name and network address, as follows:
`user_name/password@SCAN/service_name`. Server-side work such as Oracle Scheduler, Parallel Query, and Oracle Streams queues set the service name as part of the

workload definition. For Oracle Scheduler, jobs are assigned to job classes and job classes execute within services. For Parallel Query and Parallel DML, the query coordinator connects to a service and the parallel query slaves inherit the service for the duration of the parallel execution. For Oracle Streams, streams queues are accessed using services. Work executing under a service inherits the thresholds and attributes for the service and is measured as part of the service.

Oracle Database Resource Manager binds services to consumer groups and priorities. This allows the database to manage the services in the order of their importance. For example, the DBA can define separate services for high priority online users and lower priority internal reporting applications. Likewise, the DBA can define Gold, Silver and Bronze services to prioritize the order in which requests are serviced for the same application. When planning the services for a system, the plan should include the priority of each service relative to the other services. In this way, Oracle Database Resource Manager can satisfy the priority-one services first, followed by the priority-two services, and so on.

When users or applications connect to a database, Oracle recommends that you use a service specified in the `CONNECT_DATA` portion of the connect string. Oracle Database automatically creates one database service when the database is created but the behavior of this service is different from that of database services that you subsequently create. To enable more flexibility in the management of a workload using the database, Oracle Database enables you to create multiple services and specify on which instances (or in which server pools) the services start. If you are interested in greater workload management flexibility, then continue reading this chapter to understand the added features that you can use with services.

 **Note:**

The features discussed in this chapter do not work with the following default database services: `DB_NAME`, `DB_UNIQUE_NAME`, `PDB_NAME`, `SYS$BACKGROUND`, and `SYS$USERS`. Oracle strongly recommends that you do not use these services for applications to connect to the database. You must create cluster managed services to take advantage of these features. You can only manage the services that you create. Any service that the database create automatically is managed by the database server.

Dynamic Database Services

Dynamic database services enable you to manage workload distributions to provide optimal performance for users and applications. Dynamic database services offer the following features:

- **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, route work to the optimal instances that are assigned to offer the service, and achieve high availability for planned and unplanned actions.
- **High Availability Framework:** An Oracle RAC component that enables Oracle Database to always maintain components in a running state.
- **Fast Application Notification (FAN):** Provides information to Oracle RAC applications and clients about cluster state changes and Load Balancing Advisory events, such as `UP` and `DOWN` events for instances, services, or nodes. FAN has two

methods for publishing events to clients, the Oracle Notification Service daemon, which is used by Java Database Connectivity (JDBC) clients including the Oracle Application Server, and Oracle Streams Advanced Queueing, which is only used by previous releases of Oracle Call Interface (OCI) and Oracle Data Provider for .NET (ODP.NET) clients.

**Note:**

All clients beginning with Oracle Database 12c release 2 (12.2) use Oracle Notification Service.

- **Transaction Guard:** A tool that provides a protocol and an API for at-most-once execution of transactions in case of unplanned outages and duplicate submissions.
- **Application Continuity:** Provides a general purpose infrastructure that replays the in-flight request when a recoverable error is received, masking many system, communication, and storage outages, and hardware failures. Unlike existing recovery technologies, this feature attempts to recover the transactional and non-transactional session states beneath the application, so that the outage appears to the application as a delayed execution.
- **Connection Load Balancing:** A feature of Oracle Net Services that balances incoming connections across all of the instances that provide the requested database service.
- **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 management policy that you have defined for that service. Load balancing advisory events are published through Oracle Notification Service.
- **Automatic Workload Repository (AWR):** Tracks service-level statistics as [metrics](#). Server generated alerts can be created for these metrics when they exceed or fail to meet certain thresholds.
- **Fast Connection Failover (FCF):** 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.
- **Single Client Access Name (SCAN):** Provides a single name to the clients connecting to Oracle RAC that does not change throughout the life of the cluster, even if you add or remove nodes from the cluster. Clients connecting with SCAN can use a simple connection string, such as a thin JDBC URL or EZConnect, and still achieve the load balancing and client connection failover.

You can deploy Oracle RAC and noncluster Oracle database environments to use dynamic database service 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 are described in this chapter.

Overview of Server Pools and Policy-Managed Databases

Server pools are the basis for policy-managed databases.

You can create Oracle RAC databases, whether multinode or Oracle Real Application Clusters One Node (Oracle RAC One Node), using the following deployment models:

- Administrator-managed deployment is based on the Oracle RAC deployment types that existed before Oracle Database 11g release 2 (11.2) and requires that you statically configure each database instance to run on a specific node in the cluster, and that you configure database services to run on specific instances belonging to a certain database using the `preferred` and `available` designation.
- Policy-managed deployment is based on *server pools*, where database services run within a server pool as *singleton* or *uniform* across all of the servers in the server pool. Databases are deployed in one or more server pools and the size of the server pools determine the number of database instances in the deployment.

Introduction to Server Pools

Server pools logically apportion a cluster into groups of servers offering database or application services.

Server pool properties control the scalability and availability of those databases and applications. You can configure each server pool with a minimum and maximum size, which determines scalability. Oracle Clusterware manages availability between server pools, and you can further regulate availability by configuring the importance value of individual server pools.

Servers are not assigned to server pools by name but by number. Therefore, you must configure any server to run any database. If you cannot configure servers due to, for example, heterogeneous servers or storage connectivity, then you can restrict servers by using server category definitions to determine server pool membership eligibility.

Examples of Using Server Pools

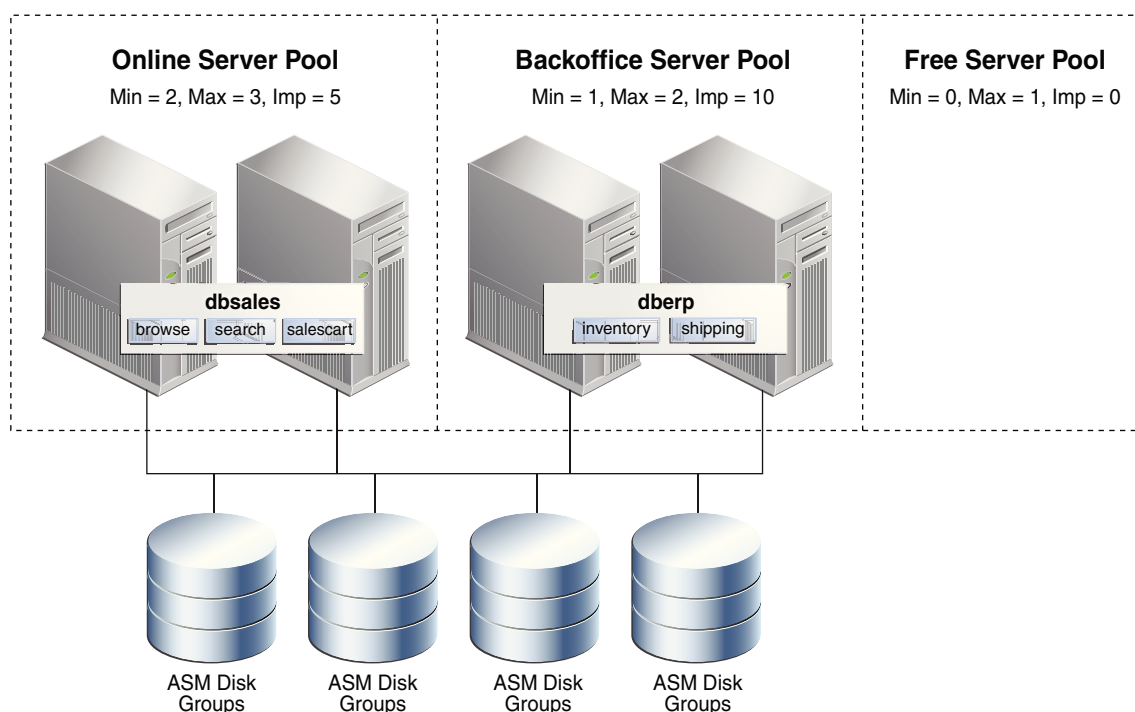
This section includes the following examples of using server pools:

- [Minimum and Maximum Number of Servers](#)
- [IMPORTANCE Attribute of Server Pools](#)
- [Consolidation of Databases](#)

Minimum and Maximum Number of Servers

Consider a four-node cluster configured into two server pools named `online` and `backoffice`. A database named `dbsales` runs in the `online` server pool offering the `browse`, `search`, and `salescart` services. A database named `dberp` runs in the `backoffice` server pool and offers the `inventory` and `shipping` services, as shown in [Figure 1-2](#). During normal business hours the enterprise requires a minimum of two instances of the `dbsales` database and one instance of the `dberp` database to meet normal demand.

Figure 1-2 Server Placement by Minimum and Maximum Limits



In this policy-managed deployment, the value of the `MIN_SIZE` server pool attribute for the `online` server pool is 2, while the value of the `MIN_SIZE` server pool attribute for the `backoffice` server pool is 1. Configured this way, Oracle Clusterware ensures that there are always two servers in the `online` server pool and one server in the `backoffice` server pool. Because this is a four-node cluster, there is one server left not assigned to either server pool. Where that last server gets deployed is determined by the `MAX_SIZE` server pool parameter of each server pool. If the sum of the values of the `MAX_SIZE` server pool attribute for each server pool is less than the total number of servers in the cluster, then the remaining servers stay in the Free server pool awaiting a failure of a deployed node.

If the value of `MAX_SIZE` is greater than that of `MIN_SIZE`, then the remaining server will be deployed into the server pool whose importance value is greatest, as shown in Figure 1-2, and fully discussed in the next section. In this case, the server is a shareable resource that can be relocated online to join the server pool where it is required. For example, during business hours the server could be given to the `online` server pool to add an instance of the `dbsales` database but after hours could be relocated to the `backoffice` server pool, adding a `dberp` database instance. All such movements are online and instances are shut down, transactionally.

These two policy-managed databases are running only the instances that are required and they can be dynamically increased or decreased to meet demand or business requirements.

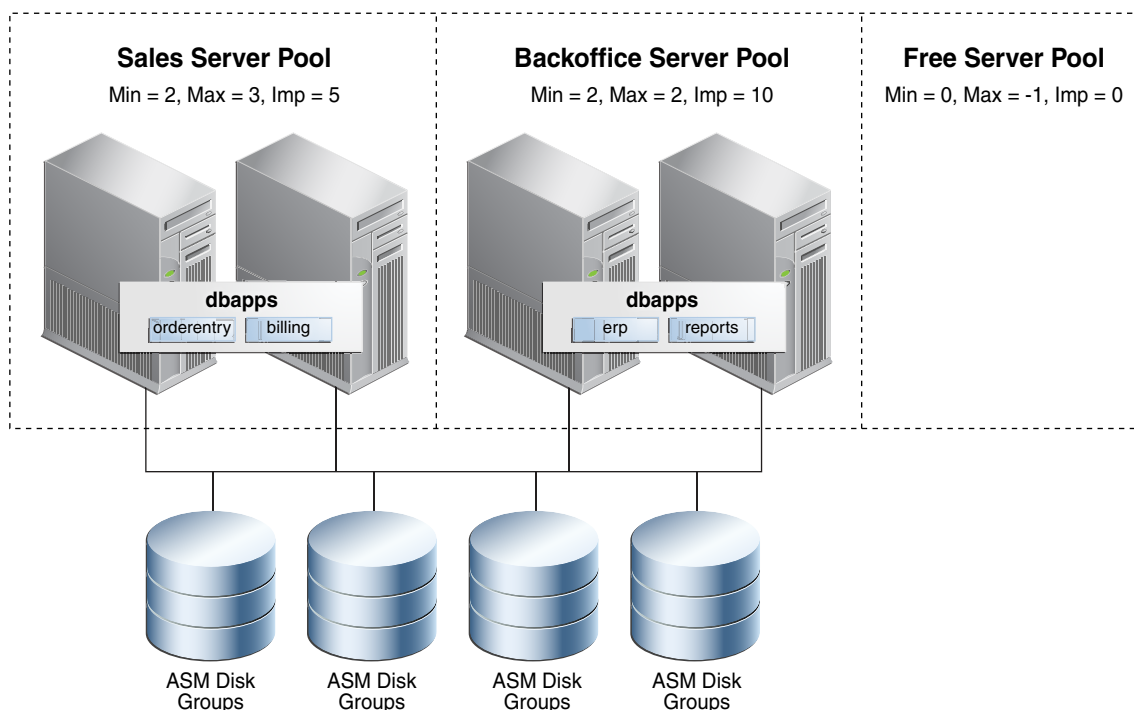
IMPORTANCE Attribute of Server Pools

The `IMPORTANCE` server pool attribute is used at cluster startup and in response to a node failure or eviction. In contrast to administrator-managed databases, you can configure server pools with different importance levels to determine which databases

are started first and which databases remain online in case there is a multinode outage.

Consider a four-node cluster that hosts a database named `dbapps` in two server pools, `sales` and `backoffice`. Two services, `orderentry` and `billing`, run in the `sales` server pool, while two other services, `erp` and `reports`, run in the `backoffice` server pool, as shown in Figure 1-3. By configuring the value of the `IMPORTANCE` server pool attribute of the `sales` server pool higher than that of the `backoffice` server pool, the services in `sales` start first when the cluster starts and are always available, even if there is only one server left running after a multinode failure. The `IMPORTANCE` server pool attribute enables you to rank services and also eliminates the requirement to run a service on all nodes in a cluster to ensure that it is always available.

Figure 1-3 Server Pool Importance



Consolidation of Databases

You can use several different approaches, either discretely or combined, to consolidate Oracle databases. Policy-managed deployments facilitate consolidation. In the case of schema consolidation, where multiple applications are being hosted in a single database separated into discrete schemas or pluggable databases (PDBs), you can use server pools to meet required capacity. Because of the dynamic scaling property of server pools, you can increase or decrease the number of database instances to meet current demand or business requirements. Since server pools also determine which services run together or separately, you can configure and maintain required affinity or isolation.

When it is not possible to use schema consolidation because of, for example, version requirements, you can host multiple databases on a single set of servers. Using policy-managed databases facilitates this database consolidation because they can share the same server pool by making use of [instance caging](#), which enables you to

dynamically increase or decrease databases, both horizontally (using server pool size) and vertically (using the `CPU_COUNT` server configuration attribute) to meet demand or business policies and schedules.

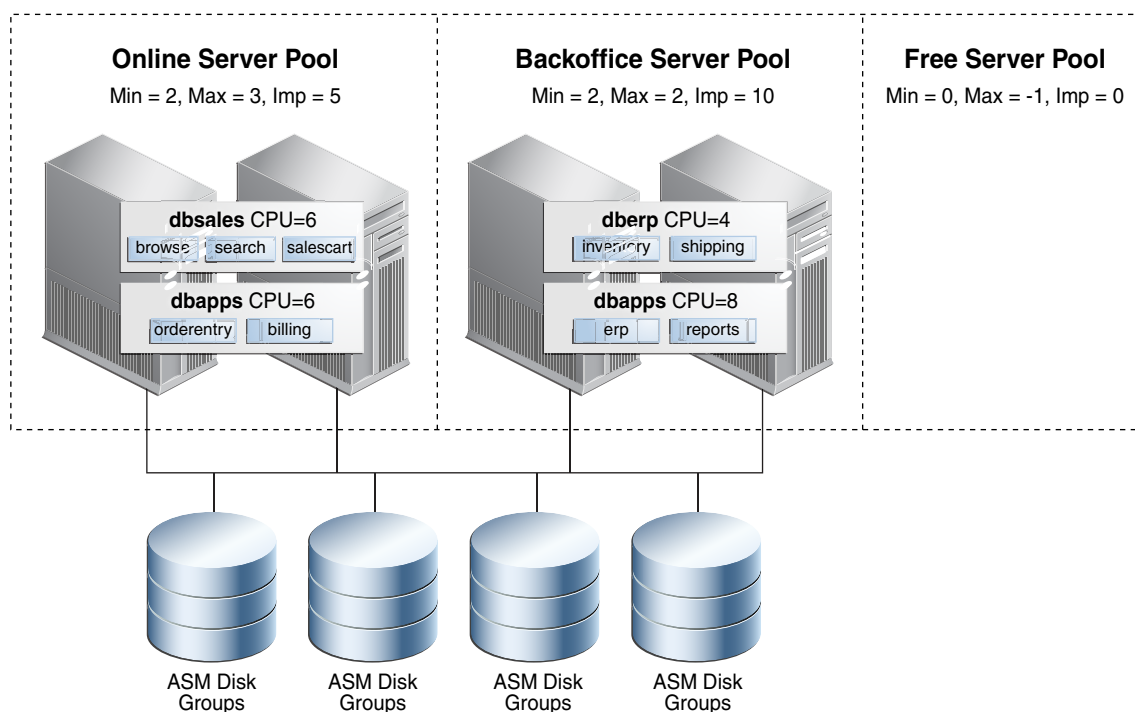
By contrast, with administrator-managed databases, you are required to reserve capacity on each server to absorb workload failing over should a database instance or server fail. With policy-managed databases, however, you can effectively rank server pools by the business necessity of the workloads that they are running using the `MIN_SIZE`, `MAX_SIZE`, and `IMPORTANCE` server pool attributes.

When the failure of a server brings a server pool to below its configured minimum number of servers, another server will move from a less important server pool to take its place and bring the number of servers back up to the configured minimum. This eliminates the risk of cascade failures due to overloading the remaining servers and enables you to significantly reduce or even eliminate the need to reserve capacity for handling failures.

Migrating or converting to policy-managed databases also enables cluster consolidation and creates larger clusters that have greater availability and scalability because of the increased number of servers available to host and scale databases. Because policy-managed databases do not require binding their instance names to a particular server and binding services to particular instances, the complexity of configuring and managing large clusters is greatly reduced.

An example deployment is shown in [Figure 1-4](#) where the previous two cluster examples (shown in [Figure 1-2](#) and [Figure 1-3](#)) are consolidated into a single cluster, making use of both database consolidation (using instance caging) and cluster consolidation (using server pools) configured so that workloads are properly sized and prioritized.

Figure 1-4 Consolidating Databases



Deploying Policy-Managed Databases

When you deploy a policy-managed database you must first determine the services and their required sizing, taking into account that services cannot span server pools.

If you are going to collocate this database with other databases, then you should factor in its CPU requirements relative to the other hosted databases, and also factor in the value of its `CPU_COUNT` attribute for instance caging, so that you can size the database both vertically and horizontally in one or more server pools.

If you are going to collocate the server pools for this database with other server pools, then consider configuring the server pools to adjust the server pool sizes on a calendar or event basis to optimize meeting demand and business requirements. Once you have determined the sizes of the server pools, and configured the appropriate values for the `MIN_SIZE` and `MAX_SIZE` server pool attributes, you can then determine the relative importance of each server pool.

You, as the cluster administrator, create policy-managed database server pools using the `srvctl add serverpool` command. You can modify the properties of the server pool using the `srvctl modify serverpool` command in the Oracle Grid Infrastructure home.

While it is possible to create a server pool using DBCA, Oracle recommends this only for small, single server pool deployments, because DBCA will fail if servers are already allocated to other server pools. Additionally, if the cluster is made up of servers with different capacities, such as old and new servers, Oracle recommends that you set up server category definitions defining the minimum server requirements for a server to join each server pool.

After you create the server pools, you can run DBCA from the appropriate database home. Depending on the database type and task, you will be presented with different default options. For all new Oracle RAC and Oracle RAC One Node databases, including container databases (CDBs), the Policy-Managed option is the default and the option that Oracle recommends.

If you are upgrading your database from an administrator-managed database or a database earlier than Oracle Database 11g release 2 (11.2), then you will not have the option to directly upgrade to a policy-managed database. After you upgrade, however, you can convert the database to policy managed using the `srvctl modify database` command.

When you convert from an administrator-managed database to a policy-managed database, the instance names are automatically updated to include the underscore (for example: `orcl1` becomes `orcl_1`). The underscore is required so that the database can automatically create instances when a server pool grows in size.

Managing Policy-Managed Databases

Managing a policy-managed database requires less configuration and reconfiguration steps than an administrator-managed one with respect to creation, sizing, patching, and load balancing.

Also, because any server in the server pools within the cluster can run any of the databases, you do not have to create and maintain database instance-to-node-name mappings. If, however, you want a database to use a specific instance name whenever it runs on a particular node, then you can create instance-to-node-name mappings using the `srvctl modify instance -db db_unique_name -instance inst_name -`

`node node_name` command. This can be useful when scripts on a particular node connect to the database using a fixed `ORACLE_SID` value.

You can perform maintenance tasks such as patching by relocating servers into the Free pool or by adjusting the server pool minimum and maximum sizes, thereby retaining required availability.

Policy-managed databases also facilitate the management of services, because they are assigned to a single server pool and run as singletons or uniform across all servers in the pool. You no longer have to create or maintain explicit preferred and available database instance lists for each service. If a server moves into a server pool because of manual relocation or a high availability event, all uniform services and their dependent database instances are automatically started. If a server hosting one or more singleton services goes down, those services will automatically be started on one or more of the remaining servers in the server pool. In the case of Oracle RAC One Node, the corresponding database instance will also be started automatically.

Managing services relative to each other is improved by making use of the importance attribute of each server pool. Each service running in a server pool inherits the server pool's importance relative to the other server pool-hosted services in the cluster. If the minimum size of the most important server pool is greater than zero, then the services and associated database instances in that server pool are started first on cluster startup and will be the last services and database instances running, as long as there is one server running in the cluster. You can offer services not critical to the business in the least important server pool, ensuring that, should sufficient resources not be available due to demand or failures, those services will eventually be shut down and the more business-critical services remain available.

Because many management tasks may involve making changes that can affect multiple databases, services, or server pools in a consolidated environment, you can use the evaluate mode for certain `SRVCTL` commands to get a report of the resource impact of a command.

Consider the following example, that evaluates the effect on the system of modifying a server pool:

```
$ srvctl modify srvpool -l 3 -g online -eval

Service erp1 will be stopped on node test3
Service reports will be stopped on node test3
Service inventory will be stopped on node test3
Service shipping will be stopped on node test3
Database dbsales will be started on node test3
Service orderentry will be started on node test3
Service billing will be started on node test3
Service browse will be started on node test3
Service search will be started on node test3
Service salescart will be started on node test3
Server test3 will be moved from pool backoffice to pool online
```

As shown in the preceding example, modifying a server pool can result in many resource state changes. You can use a *policy set* through either Oracle Clusterware or Oracle Database Quality of Service Management.

Policy-Based Cluster Management

Oracle Clusterware supports the management of a cluster configuration policy set as a native Oracle Clusterware feature.

A [cluster configuration policy](#) contains one definition for each server pool that is defined in the system. A cluster configuration policy also specifies resource placement and cluster node availability. A [cluster configuration policy set](#) defines the names of all of the server pools that are configured in a cluster, and contains one or more configuration policies.

There is always only one configuration policy in effect at any one time. However, administrators typically create several configuration policies to reflect the different business needs and demands based on calendar dates or time of day parameters. For instance, morning hours during business days are typically when most users log in and download their email; email-related workloads are usually light at nighttime and on weekends. In such cases, you can use cluster configuration policies to define the server allocation based on the expected demand. More specifically for this example, a configuration policy that allocates more servers to OLTP workloads is in effect during workday mornings, and another configuration policy allocates more servers to batch workloads on weekends and workday evenings.

Using cluster configuration policies can also help manage clusters that comprise servers of different capabilities, such as different computer and memory sizes (heterogeneous). To create management and availability policies for clusters comprised of heterogeneous server types, the cluster administrator can create server categories based on server attributes. These attributes can restrict which servers can be assigned to which server pools. For example, if you have some servers in a cluster that run older hardware, then you can use an attribute to specify that these servers should only be allocated to the server pools that support batch jobs and testing, instead of allocating them to the server pools that are used for online sales or other business-critical applications.

Overview of Oracle Database Quality of Service Management

Oracle Database Quality of Service Management (Oracle Database QoS Management) is an automated, policy-based product that monitors the workload requests for an entire system.

Oracle Database QoS Management manages the resources that are shared across applications, and adjusts the system configuration to keep the applications running at the performance levels needed by your business. Oracle Database QoS Management responds gracefully to changes in system configuration and demand, thus avoiding additional oscillations in the performance levels of your applications.

Oracle Database QoS Management monitors and manages Oracle RAC database workload performance objectives by identifying bottlenecked resources impacting these objectives, and both recommending and taking actions to restore performance. Administrator-managed deployments bind database instances to nodes but policy-managed deployments do not, so the Oracle Database QoS Management server pool size resource control is only available for the latter. All other resource management controls are available for both deployments.

Oracle Database QoS Management supports administrator-managed Oracle RAC and Oracle RAC One Node databases with its Measure-Only, Monitor, and Management modes. This enables schema consolidation support within an administrator-managed Oracle RAC database by adjusting the CPU shares of performance classes running in the database. Additionally, database consolidation is supported by adjusting CPU counts for databases hosted on the same physical servers.

Because administrator-managed databases do not run in server pools, the ability to expand or shrink the number of instances by changing the server pool size that is supported in policy-managed database deployments is not available for administrator-managed databases. This new deployment support is integrated into the Oracle QoS Management pages in Oracle Enterprise Manager Cloud Control.

Overview of Hang Manager

Hang Manager is an Oracle Database feature that automatically detects and resolves system hangs.

Hang Manager was first available in Oracle Database 11g release 1 (11.1) and, initially, identified system hangs and then dumped the relevant information about the hang into a trace file. In Oracle Database 12c release 2 (12.2), Hang Manager can take action on and attempt to resolve the system hang. Hang Manager also runs in both single-instance and Oracle RAC database instances.

Note:

Hang manager does not resolve dead locks, although dead lock detection has been available in Oracle Database for several of the past releases.

Hang Manager functions, as follows:

- First detects a system hang and then analyzes the hang and verifies the cause of the hang. It then applies heuristics to decide on a course of action to resolve the hang.
- Automates the tasks that used to require manual steps by a DBA to provide the trace files to My Oracle Support so that someone there could identify the source of the hang, minimizing or eliminating database and application downtime.
- Periodically scans all processes and analyzes a smaller subset of processes that are holding resources in successive scans. Hang manager ignores processes if there is nothing waiting on the resource.
- Considers cross-instance hangs, which are hangs where the holder is a database process waiting on a response from an Oracle ASM instance.
- Is aware of processes running in reader nodes instances, and checks whether any of these processes are blocking progress on Hub Nodes and takes action, if possible.
- Considers the Oracle Database Quality of Service Management settings of the holder.
- Terminates the holder process so the next process waiting on that resource can move on and prevent a hang.
- Notifies a DBA with an ORA-32701 error message in the alert log.

See Also:

Oracle Autonomous Health Framework User's Guide

Overview of Oracle Multitenant with Oracle RAC

You can configure a multitenant container database (CDB) to use Oracle RAC.

You can make each PDB available on either every database instance of the Oracle RAC CDB or on a subset of instances. In either case, access to PDBs is regulated using dynamic database services. Applications use these services to connect to a PDB, just as they would connect to a single-instance non-CDB.

You can isolate PDBs to prevent certain operations from being performed on or within a particular PDB that may interfere with other PDBs sharing the same Oracle RAC database or instance. PDB isolation allows for greater consolidation.

If you create an Oracle RAC database as a CDB, and if you plug PDBs into the CDB, then by default a PDB is not started automatically on any instance. With the first dynamic database service assigned to the PDB (other than the default database service, which has the same name as the database name), the PDB is made available on those instances on which the service runs.

Regardless of whether a PDB is available on multiple instances of an Oracle RAC CDB, the CDB is typically managed by the services running on the PDB. You can manually enable PDB access on each instance by starting the PDB manually on that instance.

Overview of Database In-Memory and Oracle RAC

Each node in an Oracle RAC environment has its own In-Memory (IM) column store. By default, populated objects are distributed across all IM column stores in the cluster.

Each node in an Oracle RAC environment has its own In-Memory (IM) column store. Oracle recommends that you equally size the IM column stores on each Oracle RAC node. For any Oracle RAC node that does not require an IM column store, set the `INMEMORY_SIZE` parameter to 0.

It is possible to have completely different objects populated on every node, or to have larger objects distributed across all of the IM column stores in the cluster. It is also possible to have the same objects appear in the IM column store on every node (on engineered systems, only). The distribution of objects across the IM column stores in a cluster is controlled by two additional sub-clauses to the `INMEMORY` attribute; `DISTRIBUTE` and `DUPLICATE`.

In an Oracle RAC environment, an object that only has the `INMEMORY` attribute specified on it is automatically distributed across the IM column stores in the cluster. The `DISTRIBUTE` clause can be used to specify how an object is distributed across the cluster. By default, the type of partitioning used (if any) determines how the object is distributed. If the object is not partitioned it is distributed by rowid range. Alternatively, you can specify the `DISTRIBUTE` clause to over-ride the default behavior.

On an Engineered System, it is possible to duplicate or mirror objects populated in memory across the IM column store in the cluster. This provides the highest level of redundancy. The `DUPLICATE` clause is used to control how an object should be duplicated across the IM column stores in the cluster. If you specify just `DUPLICATE`, then one mirrored copy of the data is distributed across the IM column stores in the

cluster. If you want to duplicate the entire object in each IM column store in the cluster, then specify `DUPLICATE ALL`.

 **Note:**

When you deploy Oracle RAC on a non-Engineered System, the `DUPLICATE` clause is treated as `NO DUPLICATE`.

Overview of Managing Oracle RAC Environments

This section describes the following Oracle RAC environment management topics:

- [About Designing and Deploying Oracle RAC Environments](#)
- [About Administrative Tools for Oracle RAC Environments](#)
- [About Monitoring Oracle RAC Environments](#)
- [About Evaluating Performance in Oracle RAC Environments](#)

About Designing and Deploying Oracle RAC 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 achieves both business and technical objectives.

About Administrative Tools for Oracle RAC Environments

You administer a cluster database as a single-system image using the Server Control Utility (SRVCTL), Oracle Enterprise Manager, SQL*Plus, and other utilities.

- **Server Control Utility (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 Real Application Clusters One Node (Oracle RAC One Node), Oracle Clusterware, and Oracle ASM.
- **Rapid Home Provisioning:** Use Rapid Home Provisioning to patch, upgrade, and provision Oracle RAC databases.
- **Oracle Enterprise Manager:** Oracle Enterprise Manager Cloud Control GUI interface for managing both noncluster database and Oracle RAC database environments. Oracle recommends that you use Oracle Enterprise Manager to perform administrative tasks whenever feasible.

You can use Oracle Enterprise Manager Cloud Control to also manage Oracle RAC One Node databases.

- **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, in addition to operating system groups and users. You can use CVU for preinstallation checks and for postinstallation checks of your cluster environment. CVU is especially useful during preinstallation and during installation of Oracle Clusterware and Oracle RAC components. Oracle Universal Installer 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 CVU for verifying the completion of ongoing administrative tasks, such as node addition and node deletion.

- **DBCA:** The recommended utility for creating and initially configuring Oracle RAC, Oracle RAC One Node, and Oracle noncluster databases.
- **NETCA:** Configures the network for your Oracle RAC environment.

About Monitoring Oracle RAC Environments

Web-based Oracle Enterprise Manager Cloud Control enables you to monitor an Oracle RAC database.

Oracle Enterprise Manager Cloud Control is a central point of control for the Oracle environment that you access by way of a graphical user interface (GUI). See "Monitoring Oracle RAC and Oracle Clusterware" and *Oracle Database 2 Day + Real Application Clusters Guide* for detailed information about using Oracle Enterprise Manager to monitor Oracle RAC environments.

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

- Use Oracle Enterprise Manager Cloud Control to initiate cluster database management tasks.
- Use Oracle Enterprise Manager Cloud Control to administer multiple or individual Oracle RAC databases.
- Use the global views (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.

For almost every V\$ view, there is a corresponding global GV\$ view. In addition to the V\$ information, each GV\$ view contains an extra column named `INST_ID`, which displays the instance number from which the associated V\$ view information was obtained.

- Use the sophisticated management and monitoring features of the Oracle Database Diagnostic and Tuning packs within Oracle Enterprise Manager that include the Automatic Database Diagnostic Monitor (ADDM) and Automatic Workload Repository (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.

About Evaluating Performance in Oracle RAC Environments

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

Some of the performance features specific to Oracle RAC include:

- Dynamic resource allocation
 - Oracle Database 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
 - Oracle RAC-specific performance views in Oracle Enterprise Manager

2

Administering Storage in Oracle RAC

Oracle recommends Oracle Automatic Storage Management (Oracle ASM) as a storage management solution that provides an alternative to conventional volume managers, file systems, and raw devices.

Oracle ASM is a [volume manager](#) and a [file system](#) for Oracle database files that supports single-instance Oracle Database and Oracle Real Application Clusters (Oracle RAC) configurations.

Oracle ASM uses disk groups to store data files; an Oracle ASM disk group is a collection of disks that Oracle ASM manages as a unit. Within a disk group, Oracle ASM exposes a file system interface for Oracle database files. The content of files that are stored in a disk group is evenly distributed to eliminate hot spots and to provide uniform performance across the disks. The performance is comparable to the performance of raw devices.

You can add or remove disks from a disk group while a database continues to access files from the disk group. When you add or remove disks from a disk group, Oracle ASM automatically redistributes the file contents and eliminates the need for downtime when redistributing the content.

The Oracle ASM volume manager functionality provides flexible server-based mirroring options. The Oracle ASM normal and high redundancy disk groups enable two-way and three-way mirroring respectively. You can use external redundancy to enable a Redundant Array of Independent Disks (RAID) storage subsystem to perform the mirroring protection function.

Oracle ASM also uses the Oracle Managed Files feature to simplify database file management. Oracle Managed Files automatically creates files in designated locations. Oracle Managed Files also names files and removes them while relinquishing space when tablespaces or files are deleted.

Oracle ASM reduces the administrative overhead for managing database storage by consolidating data storage into a small number of disk groups. The smaller number of disk groups consolidates the storage for multiple databases and provides for improved I/O performance.

Oracle ASM files can coexist with other storage management options such as raw disks and third-party file systems. This capability simplifies the integration of Oracle ASM into pre-existing environments.

Oracle ASM has easy to use management interfaces such as SQL*Plus, the Oracle ASM Command Line Utility (ASMCMD) command-line interface, and Oracle ASM Configuration Assistant (ASMCA).

This chapter includes the following topics:

- [Overview of Storage Management for Oracle RAC](#)
- [Data File Access in Oracle RAC](#)
- [NFS Server for Storage](#)

- [Redo Log File Storage in Oracle RAC](#)
- [Automatic Undo Management in Oracle RAC](#)
- [Oracle Automatic Storage Management with Oracle RAC](#)

Overview of Storage Management for Oracle RAC

All data files (including an undo tablespace for each instance) and redo log files (at least two for each instance) for an Oracle RAC database must reside on shared storage.

Oracle recommends that you use Oracle ASM to store these files in an Oracle ASM disk group.

Oracle supports alternative ways of using shared storage, such as certified [cluster file systems](#). In addition, Oracle recommends that you use one shared server parameter file (SPFILE) with instance-specific entries. Oracle RAC 12c allows storing shared password files in Oracle ASM and storing Oracle Database files on Oracle Automatic Storage Management Cluster File System (Oracle ACFS).



Note:

Oracle Database and related technologies, such as Oracle Clusterware, no longer support the use of raw (block) storage devices. You must move files to Oracle ASM before upgrading to Oracle Clusterware 12c.

Unless otherwise noted, Oracle Database storage features such as Oracle ASM, Oracle Managed Files, automatic segment-space management, and so on, function the same in Oracle RAC environments as they do in noncluster Oracle database environments.



Note:

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

Data File Access in Oracle RAC

All Oracle RAC instances must be able to access all data files. If a data file must 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 data files. Similarly, when you add a tablespace or data file or bring a tablespace or data file online, all instances verify access to the file or files.

If you add a data file to a disk that other instances cannot access, then verification fails. Verification also fails if instances access different copies of the same data file. 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 data file access.

NFS Server for Storage

An Oracle database can serve as a network file system (NFS) server. The database responds to NFS requests from any NFS client and stores both the files and their metadata within the database.

Files associated with a primary database, such as SQL scripts, can be automatically replicated on a standby database. You can also store unstructured data, such as emails, on the database.

You can create or destroy an Oracle file system and access it through the NFS server using the procedures documented in *Oracle Database SecureFiles and Large Objects Developer's Guide*.

Redo Log File Storage in Oracle RAC

In an Oracle RAC database, 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 you use DBCA to create the database, DBCA allocates redo log files to instances, as required, automatically. You can change the number of redo log groups and the size of the redo log files as required either during the initial database creation or as a post-creation step.

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, then you should disable the instance's thread of redo so that Oracle does not have to check the thread during database recovery.

Redo log management must be considered when the number of instances for a particular production Oracle RAC database changes. For example, if you increase the cardinality of a server pool for a policy-managed database and a new server is allocated to the server pool, then Oracle starts an instance on the new server. As soon as the database instance on the new server starts, it will require a set of redo log groups. Using Oracle Managed Files based on an Oracle ASM disk group, allocation of the required redo log thread and respective files is performed automatically. You should create redo log groups only if you use administrator-managed databases.

For administrator-managed databases, each instance has its own online redo log groups. Create these redo log groups and establish group members. To add a redo log group to a specific instance, specify the `INSTANCE` clause in the `ALTER DATABASE ADD LOGFILE` statement. If you do not specify the instance when adding the redo log group, then 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, then you should disable the instance's thread of redo so that Oracle does not have to check the thread during database recovery.

Automatic Undo Management in Oracle RAC

Oracle Database automatically manages undo segments within a specific undo tablespace that is assigned to an instance. 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, if that undo tablespace is not currently used by another instance for undo generation or transaction recovery.

You assign undo tablespaces in your Oracle RAC administrator-managed database by specifying a different value for the `UNDO_TABLESPACE` parameter for each instance in your SPFILE or individual PFILES. For policy-managed databases, Oracle automatically allocates the undo tablespace when the instance starts if you have Oracle Managed Files enabled. 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.

Oracle Automatic Storage Management with Oracle RAC

Oracle ASM automatically maximizes I/O performance by managing the storage configuration across the disks that Oracle ASM manages.

Oracle ASM does this by evenly distributing the database files across all of the available storage assigned to the disk groups within Oracle ASM. Oracle ASM allocates your total disk space requirements into uniformly sized units across all disks in a disk group. Oracle ASM can also automatically mirror files to prevent data loss. Because of these features, Oracle ASM also significantly reduces your administrative overhead.

Oracle ASM instances are created on each node where you install Oracle Clusterware. Each Oracle ASM instance has either an SPFILE or PFILE type parameter file. Oracle recommends that you back up the parameter files and the TNS entries for nondefault Oracle Net listeners.

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

The following topics describe Oracle ASM and Oracle ASM administration, as follows:

- [Storage Management in Oracle RAC](#)
- [Modifying Disk Group Configurations for Oracle ASM](#)
- [Oracle ASM Disk Group Management](#)
- [Configuring Preferred Mirror Read Disks in Extended Distance Clusters](#)
- [Converting Nonclustered Oracle ASM to Clustered Oracle ASM](#)
- [Administering Oracle ASM Instances with SRVCTL in Oracle RAC](#)

Storage Management in Oracle RAC

You can create Oracle ASM disk groups and configure mirroring for Oracle ASM disk groups using the Oracle ASM configuration assistant (ASMCA).

Alternatively, you can use Oracle Enterprise Manager to administer Oracle ASM disk groups after you have discovered the respective servers with Oracle Enterprise Manager.

The Oracle tools that you use to manage Oracle ASM, including ASMCA, Oracle Enterprise Manager, and the silent mode install and upgrade commands, include options to manage Oracle ASM instances and disk groups.

You can use the Cluster Verification Utility (CVU) to verify the integrity of Oracle ASM across the [cluster](#). Typically, this check ensures that the Oracle ASM instances on all nodes run from the same Oracle home and, if `asmlib` exists, that it is a valid version and has valid ownership. Run the following command to perform this check:

```
cluvfy comp asm [-n node_list] [-verbose]
```

Replace `node_list` with a comma-delimited list of node names on which the check is to be performed. Specify `all` to check all nodes in the cluster.

Use the `cluvfy comp ssa` command to locate shared storage.

Modifying Disk Group Configurations for Oracle ASM

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 Oracle ASM with an Oracle RAC database compared to using it with a noncluster Oracle database. Oracle ASM automatically redistributes the data files after you add or delete a disk or disk group.

In a cluster, each Oracle ASM instance manages its node's metadata updates to the disk groups. In addition, each Oracle ASM instance coordinates disk group metadata with other nodes in the cluster. As with noncluster Oracle databases, you can use Oracle Enterprise Manager, ASMCA, SQL*Plus, and the Server Control Utility (SRVCTL) to administer disk groups for Oracle ASM that are used by Oracle RAC. *Oracle Automatic Storage Management Administrator's Guide* explains how to use SQL*Plus to administer Oracle ASM instances. Subsequent sections describe how to use the other tools.

 **Note:**

When you start ASMCA, if there is not an Oracle ASM instance, then the utility prompts you to create one.

Oracle ASM Disk Group Management

To use Oracle ASM, you must first create disk groups with ASMCA before creating a database with DBCA.

You can also use the disk group management commands to create and manage an Oracle ASM instance and its associated disk groups independently of creating a database. You can use Oracle Enterprise Manager or ASMCA to add disks to a disk group, to mount a disk group or to mount all of the disk groups, or to create Oracle ASM instances. Additionally, you can use Oracle Enterprise Manager to dismount and drop disk groups or to delete Oracle ASM instances.

Oracle ASM instances are created when you install Oracle Clusterware. To create an Oracle ASM disk group, run ASMCA from the *Grid_home/bin* directory. You can also use the Oracle ASM Disk Groups page in ASMCA for Oracle ASM management. That is, you can configure Oracle ASM storage separately from database creation. For example, from the ASM Disk Groups page, you can create disk groups, add disks to existing disk groups, or mount disk groups that are not currently mounted.

When you start ASMCA, if the Oracle ASM instance has not been created, then ASMCA prompts you to create the instance. ASMCA prompts you for the `sysasm` password and the `ASMSNMP` password.

Configuring Preferred Mirror Read Disks in Extended Distance Clusters

When you configure Oracle 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 Oracle ASM to read from a secondary extent if that extent is closer to the node instead of Oracle 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.

Converting Nonclustered Oracle ASM to Clustered Oracle ASM

When installing Oracle Grid Infrastructure, any nonclustered Oracle ASM instances are automatically converted to clustered Oracle ASM.

Administering Oracle ASM Instances with SRVCTL in Oracle RAC

You can use the Server Control Utility (SRVCTL) to add or remove an Oracle ASM instance.

To issue SRVCTL commands to manage Oracle ASM, log in as the operating system user that owns the Oracle Grid Infrastructure home and issue the SRVCTL commands from the `bin` directory of the Oracle Grid Infrastructure home.

Use the following syntax to add an Oracle ASM instance:

```
srvctl add asm
```

Use the following syntax to remove an Oracle ASM instance:

```
srvctl remove asm [-force]
```

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

Use the following syntax to start an Oracle ASM instance:

```
srvctl start asm [-node node_name] [-startoption start_options]
```

Use the following syntax to stop an Oracle ASM instance:

```
srvctl stop asm [-node node_name] [-stopoption stop_options]
```

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

```
srvctl config asm -node node_name
```

Use the following syntax to display the state of an Oracle ASM instance:

```
srvctl status asm [-node node_name]
```

3

Administering Database Instances and Cluster Databases

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

The topics in this chapter include:

- [Overview of Oracle RAC Database Administration](#)
- [Starting and Stopping Instances and Oracle RAC Databases](#)
- [Starting and Stopping PDBs in Oracle RAC](#)
- [Verifying That Instances are Running](#)
- [Terminating Sessions On a Specific Cluster Instance](#)
- [Overview of Initialization Parameter Files in Oracle RAC](#)
- [Initialization Parameter Use in Oracle RAC](#)
- [Converting an Administrator-Managed Database to a Policy-Managed Database](#)
- [Managing Memory Pressure for Database Servers](#)
- [Quiescing Oracle RAC 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:

The Oracle Enterprise Manager Cloud Control online help for more information about Oracle Enterprise Manager Cloud Control

Overview of Oracle RAC Database Administration

Oracle RAC database administration requires certain privileges and involves either a policy-managed or administrator-managed deployment model.

Required Privileges for Oracle RAC Database Administration

To increase security and further separate administrative duties, Oracle RAC database administrators manage Oracle RAC databases with the SYSRAC administrative privilege, and no longer require the SYSDBA administrative privilege. The SYSRAC administrative privilege is the default mode of connecting to the database by the Oracle Clusterware agent on behalf of Oracle RAC utilities, such as SRVCTL,

meaning that no SYSDBA connections to the database are necessary for everyday administration of Oracle RAC database clusters.

Oracle RAC Database Deployment Models

Oracle RAC databases support two different management styles and deployment models:

- Administrator-managed deployment is based on the Oracle RAC deployment types that existed before Oracle Database 11g release 2 (11.2) and requires that you statically configure each database instance to run on a specific node in the cluster, and that you configure database services to run on specific instances belonging to a certain database using the `preferred` and `available` designation.
- Policy-managed deployment is based on *server pools*, where database services run within a server pool as *singleton* or *uniform* across all of the servers in the server pool. Databases are deployed in one or more server pools and the size of the server pools determine the number of database instances in the deployment.

You can manage databases with either the administrator-managed or policy-managed deployment model using the same commands or methods (such as DBCA or Oracle Enterprise Manager). All commands and utilities maintain backward compatibility to support the management of Oracle databases that only support administrator-based management (Oracle databases before Oracle Database 11g release 2 (11.2)).

In general, a database is defined as a resource in Oracle Clusterware. The database resource is automatically created when you create your database with DBCA or provision a database using Rapid Home Provisioning, or you can manually create the database resource by adding your database with SRVCTL. The database resource contains the Oracle home, the SPFILE, one or more server pools, and one or more Oracle ASM disk groups required for the database to start. You can specify the Oracle ASM disk groups using either the `srvctl add database` or `srvctl modify database` commands, or, when the database opens a data file on a disk group that is not on this list, the disk group gets added to the list.

The database resource also has a *weak start dependency* on the listener type, which means that the resource tries to start all listeners for the node when the database instance starts. Oracle Clusterware tries to start listeners on the node where the database instance starts. Starting the listeners in turn starts the VIP for the node.

When you review the database resource for an administrator-managed database, you see a server pool defined with the same name as the Oracle database. This server pool is part of a special Oracle-defined server pool called *Generic*. Oracle RAC manages the Generic server pool to support administrator-managed databases. When you add or remove an administrator-managed database using either SRVCTL or DBCA, Oracle RAC creates or removes the server pools that are members of Generic. You cannot use SRVCTL or CRSCTL commands to modify the Generic server pool.

Use [policy-managed databases](#) to simplify management of dynamic systems. Policy management allows clusters and databases to expand or shrink as requirements change. If you use policy-managed databases, then you must install the Oracle home software on every node in your cluster. Policy-managed databases must use Oracle Database 11g release 2 (11.2) or higher software and cannot coexist on the same servers as administrator-managed databases.

 **Note:**

You cannot run more than one instance of the same database on the same node.

A policy-managed database is defined by [cardinality](#), which is the number of database instances you want running during normal operations. A policy-managed database runs in one or more database server pools that the cluster administrator creates in the cluster, and it can run on different servers at different times. Every server pool of a policy-managed database should have at least one database service. A database instance starts on a server that is in the server pools defined for the database. If you are using Oracle Automatic Storage Management (Oracle ASM) with Oracle Managed Files for your database storage, then, when an instance starts and there is no redo thread available, Oracle RAC automatically enables one and creates the required redo log files and undo tablespace. Clients can connect to a policy-managed database using the same SCAN-based connect string no matter which servers they happen to be running on at the time.

Policy-managed databases are named `db_unique_name_cardinality`, where *cardinality* is the cardinality ID of the server in the server pool. Use the `srvctl status database -sid` command to retrieve the instance name on the local node. You can also create a fixed mapping of nodes-to-instance name using the `srvctl modify instance` command.

Using the Same Cluster for Administrator-Managed and Policy-Managed Databases

If you want to create an administrator-managed database on a cluster that already hosts policy-managed databases, then you must carefully select the nodes for the administrator-managed database. This is because the nodes that you select for an administrator-managed database that are in policy-managed server pools will be moved into the Generic server pool as part of this process.

If you select nodes that already run other policy-managed database instances, then DBCA prompts you with a message that lists the instances and services that will be shut down when DBCA creates the administrator-managed database. If you select the **Yes** button on the dialog box when DBCA asks "Do you want to continue?", then your policy-managed database instances and services will be shut down because of the administrator-managed database creation process.

 **Note:**

This is also true if you use the `srvctl add instance` command, which returns a similar error message indicating that the databases would be shut down. If you also use the force option (`-f`) with the `srvctl add instance` command, then this is the same as choosing **Yes** on the DBCA dialog. Doing this shuts down any policy-managed databases that are running on the node before moving the node into the Generic server pool.

Tools for Administering Oracle RAC

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

- [Administering Oracle RAC with SRVCTL](#)
- [Administering Oracle RAC with Oracle Enterprise Manager](#)
- [Administering Oracle RAC with SQL*Plus](#)

Administering Oracle RAC with SRVCTL

The Server Control Utility (SRVCTL) is a command-line interface that you can use to manage Oracle Databases in a centralized manner.

Oracle made centralized, SRVCTL-based database management available in Oracle Database 11g release 2 (11.2) for single-instance Oracle Databases, using Oracle ASM in the Oracle Grid Infrastructure, for both a noncluster environment and Oracle RAC databases, based on Oracle Grid Infrastructure for a cluster. This enables homogeneous management of all Oracle Database types using SRVCTL. 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, in addition to other resources in the cluster.

When you use SRVCTL to perform configuration operations on your cluster, SRVCTL stores configuration data in the Oracle Cluster Registry (OCR) in a cluster or Oracle Local Registry (OLR) in Oracle Restart environments. SRVCTL performs other operations, such as starting and stopping instances, by configuring and managing Oracle Clusterware resources, which define agents that perform database startup and shutdown operations using Oracle Call Interface APIs.



Note:

If you require your database (or database instance) to start using certain environment variables, then use the `srvctl setenv` command to set those variables for the database profile that is maintained for the database using SRVCTL. You do not need to set the `ORACLE_HOME` and `ORACLE_SID` environment variables, because SRVCTL maintains and sets those parameters, automatically.

Administering Oracle RAC with Oracle 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.

Based on the Oracle Enterprise Manager Cloud Control (Grid Control in Oracle Enterprise Manager 11g) graphical user interface (GUI), you can manage both non-clustered and Oracle RAC environments.

In Oracle 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 Oracle Enterprise Manager to start, stop, and monitor databases, cluster database instances, and their listeners, and 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 Oracle Enterprise Manager to manage schemas, security, and cluster database storage features.

Administering Oracle RAC with SQL*Plus

Unlike SRVCTL or Oracle Enterprise Manager, SQL*Plus is an instance-oriented management tool.

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. For an Oracle RAC environment that runs multiple instances on one database at the same time, this implies that you need to consider the extent to which SQL*Plus can operate on this instance. Due to those restrictions, you should not use SQL*Plus to manage policy-managed databases.

For example, when using pluggable databases (PDBs)—regardless of whether those databases are managed in an administrator-managed or a policy-managed style—you must consider that any alteration performed on the PDB using a SQL*Plus connection will, by default, only affect the current instance. To make changes affecting all instances that belong to the PDB, you must use the `ALTER PLUGGABLE DATABASE` command with `instance=all`. When using PDBs you must connect, using a dynamic database service (`net_service_name`), to an instance, as PDBs represent themselves as dynamic database services associated with one or more instances of an Oracle RAC database.

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.

Since the SQL*Plus prompt does not identify the current instance by default, 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, where `password` is the password:

```
CONNECT user_name@net_service_name
Enter password: password
```

Connecting as `SYSOPER` or `SYSRAC` 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 `SYSRAC` privilege to connect to and administer an Oracle ASM instance. If you use the `SYSRAC` privilege to connect to an Oracle ASM instance, then Oracle Database writes warnings to the alert log files because commands that run using the `SYSRAC` privilege on an Oracle ASM instance are deprecated.

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 noncluster 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 <code>CONNECT</code> command.
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	Displays parameter and SGA information from the current instance.
and	
SHOW SGA	

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

SQL*Plus Command	Associated Instance
STARTUP	Always affects the current instance. These are privileged SQL*Plus commands.
and	
SHUTDOWN	

Starting and Stopping Instances and Oracle RAC Databases

You can start and stop instances with Oracle Enterprise Manager, SQL*Plus, or SRVCTL.

Both Oracle Enterprise Manager and SRVCTL provide options to start and stop all of the instances in an Oracle RAC database with a single step.

Using any tool, you can choose the startup state to which you want to start the database. The state of the database and database instance will determine what operations you can perform. You can perform certain operations only when the database is in the MOUNT (NOMOUNT) state. Performing other operations requires that the database be in the OPEN state.

Note:

Oracle does not support running more than one instance of the same database on the same node.

To start an Oracle RAC database instance on a node in the cluster, you must first start the Oracle Grid Infrastructure stack on the node. An Oracle RAC database instance will not start on a server on which the Oracle Grid Infrastructure stack is not running.

Oracle Database QoS Management Policy Workload Criticality Determines Database Startup Order

If a user-created Oracle Database Quality of Service Management (Oracle Database QoS Management) policy is active, then the ranked order of the performance classes determines the order in which the associated Oracle RAC databases start or request real-time LMS process slots. Using the performance class rankings ensures that mission-critical databases running in a consolidated environment have their LMS processes run in real time, thus eliminating a resource bottleneck within inter-node communication. Because the Oracle Database QoS Management policy specifies the rank of each workload, using the value of `Max(Ranks)` for each database provides a consistent expression of the expressed business criticality of each database.

The procedures in the following sections discuss starting and stopping Oracle RAC database instances:

- [Starting One or More Instances and Oracle RAC Databases Using SRVCTL](#)
- [Stopping One or More Instances and Oracle RAC Databases Using SRVCTL](#)

- [Stopping All Databases and Instances Using CRSD](#)
- [Starting and Stopping Individual Instances Using SQL*Plus](#)

Starting One or More Instances and Oracle RAC Databases Using SRVCTL

Use SRVCTL start Oracle RAC databases and instances.



Note:

This section assumes that you are using an SPFILE for your database.

Enter the following SRVCTL syntax from the command line, providing the required database name and instance name, or include multiple instance names to start multiple specific instances:

- 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 -db db_unique_name [-startoption start_options]
```

```
$ srvctl stop database -db db_unique_name [-o stop_options]
```

The following SRVCTL command, for example, mounts all of the non-running instances of an Oracle RAC database:

```
$ srvctl start database -db orcl -startoption mount
```

- To start administrator-managed databases, enter a comma-delimited list of instance names:

```
$ srvctl start instance -db db_unique_name -instance instance_name_list [-startoption start_options]
```

In Windows you must enclose a comma-delimited list in double quotation marks ("").

- To start policy-managed databases, enter a single node name:

```
$ srvctl start instance -db db_unique_name -node node_name [-startoption start_options]
```

Note that this command also starts all enabled and non-running services that have `AUTOMATIC` management policy, and for which the database role matches one of the service's roles.

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

```
$ srvctl stop instance -db db_unique_name [-instance "instance_name_list" | -node node_name] [-stopoption stop_options]
```

You can enter either a comma-delimited list of instance names to stop several instances or you can enter a node name to stop one instance. In Windows you must enclose a comma-delimited list in double quotation marks ("").

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 shuts down the two instances, `orcl3` and `orcl4`, on the `orcl` database using the `immediate` stop option:

```
$ srvctl stop instance -db orcl -instance "orcl3,orcl4" -stopoption immediate
```

Stopping One or More Instances and Oracle RAC Databases Using SRVCTL

Use SRVCTL to stop instances and Oracle RAC databases.

The procedure for shutting down Oracle RAC instances is identical to shutting down instances in noncluster Oracle databases, with the following exceptions:

- 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. An 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.
- Using the `SHUTDOWN TRANSACTIONAL` command with the `LOCAL` option is useful to shut down a particular Oracle RAC database instance. 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 you ran the `SHUTDOWN` command either commit or rollback, which is a valid approach, if you intend to shut down all instances of an Oracle RAC database.

Note:

`SHUTDOWN TRANSACTIONAL` and `SHUTDOWN TRANSACTIONAL LOCAL` both perform the same action on a nonclustered database but the two commands are different on an Oracle RAC database.

Use SRVCTL to shut down an Oracle RAC database or database instance. The respective SRVCTL commands (`srvctl stop database` or `srvctl stop instance`) provide shutdown options to perform an optimized transactional shutdown. Use the `TRANSACTIONAL` stop option with the `srvctl stop database` command and the `TRANSACTIONAL LOCAL` stop option with the `srvctl stop instance` command.

Stopping All Databases and Instances Using CRSCTL

When you want to stop an entire node or cluster (for maintenance purposes, for example), you run either the `crsctl stop crs` command on the node or the `crsctl stop cluster -all` command, provided you have the required cluster privileges. These commands stop all database instances running on a server or in the cluster and ensure that their state is recovered after you restart the cluster. Using CRSCTL also

enables Oracle Clusterware to relocate services and other resources that can run elsewhere.

Using either of these CRCTL commands to stop all database instances on a server or in the cluster can lead to the database instances being stopped similar to *shutdown abort*, which requires an instance recovery on startup. If you use SRVCTL to stop the database instances manually before stopping the cluster, then you can prevent a *shutdown abort*, but this requires that you manually restart the database instances after restarting Oracle Clusterware.

Starting and Stopping Individual Instances Using SQL*Plus

If you want to start or stop just one instance and you are connected to your local node, then you must 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, are associated with that same SID.

Note:

This section assumes you are using an SPFILE.

To start or shutdown your local instance, initiate a SQL*Plus session and connect with the SYSRAC 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 SYSRAC
STARTUP MOUNT
```

Note:

If you use Oracle ASM disk groups, then use the SYSASM privilege instead of the SYSRAC privilege to connect to and administer the Oracle ASM instances.

Oracle recommends that you do not use SQL*Plus to manage Oracle ASM instances in an Oracle RAC environment. Oracle Clusterware automatically manages Oracle ASM instances, as required. If manual intervention is necessary, then use respective SRVCTL commands.

You can start multiple instances from a single SQL*Plus session on one node using Oracle Net Services. Connect to each instance in turn by using a Net Services connection string, typically an instance-specific alias from your `tnsnames.ora` file.

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 SYSRAC
SHUTDOWN TRANSACTIONAL
```

 **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 a connect string that uses a TNS alias that connects to a service or an Oracle Net address that lists multiple IP addresses, then you might not be connected to the specific instance you want to shut down.

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

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

It is not possible to start or stop multiple instances, simultaneously, with 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 connects 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.

Starting and Stopping PDBs in Oracle RAC

Administering a pluggable database (PDB) involves a small subset of the tasks required to administer a non-CDB.

Administering an Oracle RAC-based multitenant container database (CDB) is similar to administering a non-CDB. The differences are that some administrative tasks apply to the entire CDB, some to the CDB root, and some to specific PDBs. In this subset of tasks, most are the same for a PDB and a non-CDB. There are some differences, however, such as when you modify the open mode of a PDB. Also, a PDB administrator is limited to managing a single PDB and is not affected by other PDBs in the CDB.

You manage PDBs in an Oracle RAC CDB by managing services. This is true regardless of whether the PDBs are policy managed or administrator managed. Assign one dynamic database service to each PDB to coordinate start, stop, and placement of PDBs across instances in a clustered container database.

For example, if you have a CDB called `raccont` with a policy-managed PDB called `spark` in a server pool called `prod`, then assign a service called `plug` to this database using the following command:

```
srvctl add service -db raccont -pdb spark -service plug -serverpool prod
```

The service `plug` is uniformly managed across all nodes in the server pool. If you want to have this service running as a singleton service in the same server pool, then use the `-cardinality singleton` parameter with the preceding command.

To open the PDB `spark`, you must start the service `plug` as follows:

```
srvctl start service -db raccont -service plug
```

To stop the service `plug`:

```
srvctl stop service -db raccont -service plug
```

The PDB `spark` remains open until you close the PDB using the SQL command `ALTER PLUGGABLE DATABASE PDB_NAME CLOSE IMMEDIATE`. You can check the status of the database using the `srvctl status service` command.

Because PDBs are managed using dynamic database services, typical Oracle RAC-based management practices apply. For this reason, if the service `plug` is in the online state when Oracle Clusterware is shut down on a server hosting this service, then the service is restored to its original state after the restart of Oracle Clusterware on this server. Thus, starting PDBs is automated as with any other Oracle RAC database.

 **Note:**

Unlike SQL*Plus, SRVCTL operates on an entire cluster database. Starting a PDB using services therefore applies to multiple instances of the clustered CDB at the same time when the service is defined to run on multiple servers simultaneously and the current status of the cluster allows for this placement.

Verifying That Instances are Running

To verify that a database instance is available, use Oracle Enterprise Manager, SRVCTL, or SQL*Plus.

- [Using SRVCTL to Verify That Instances are Running](#)
- [Using SQL*Plus to Verify That Instances are Running](#)

Using SRVCTL to Verify That Instances are Running

You can use SRVCTL to verify that instances are running on a particular database.

The following command provides an example of using SRVCTL to check the status of the database instances for the Oracle RAC database named `mail`:

```
$ srvctl status database -db mail
```

This command returns output similar to the following:

```
Instance mail1 is running on node beta1011Instance mail2 is running on node beta1010
```

Additionally, you can check whether PDBs are running in the cluster by checking the availability of their assigned services, as follows:

```
$ srvctl status service -db db_unique_name -service service_name
```

Using SQL*Plus to Verify That Instances are Running

You can use SQL*Plus to verify that database instances are running.

1. On any node, from a SQL*Plus prompt, connect to a database instance by using a Net Services connection string, typically an instance-specific alias from your `tnsnames.ora` file.

```
CONNECT /@db1 as SYSRAC
```

2. Query the `V$ACTIVE_INSTANCES` view, using the following statement:

```
CONNECT SYS/as SYSRAC
Enter password: password
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 the following table.

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 <i>host_name:instance_name</i> .

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 session active transactions are rolled back, and resources held by the session (such as locks and memory areas) are immediately released and available to other sessions.

Using the `ALTER SYSTEM KILL SESSION` statement enables you to maintain strict application service-level agreements in Oracle RAC environments. Often, the goal of a service-level agreement is to carry out 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.

Note:

You can use Application Continuity to hide the cancellation of a transaction from the user, if the application initially used an Application Continuity-enabled dynamic database service to connect to the database instance.

For a more granular approach to service-level management, Oracle recommends that you use Oracle Database Quality of Service Management (Oracle Database QoS Management) for all Oracle RAC-based databases.

To terminate sessions, follow these steps:

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

Examples of Identifying and Terminating Sessions

The following examples provide three scenarios in which a user identifies and terminates a specific session. In each example, the `SYSDBA` first queries the `GV$SESSION` view for the `SCOTT` user's session to identify the session to terminate, and then runs the `ALTER SYSTEM KILL SESSION` statement to terminate the session on the instance.

Example 3-1 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 `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 3-2 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 3-3 Using the IMMEDIATE parameter

The following example includes 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>
```

Overview of Initialization Parameter Files in Oracle RAC

When you create the database, Oracle Database creates an SPFILE in the file location that you specify. This location can be either an Oracle ASM disk group or a [cluster file system](#). 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 an SPFILE 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 Oracle Enterprise Manager or `ALTER SYSTEM SQL` statements.

When creating an SPFILE, if you include the `FROM MEMORY` clause (for example, `CREATE PFILE FROM MEMORY` or `CREATE SPFILE FROM MEMORY`), then the `CREATE` statement creates a PFILE or SPFILE using the current system-wide 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.

This section includes the following topics:

- [Setting SPFILE Parameter Values for Oracle RAC](#)
- [Parameter File Search Order in Oracle RAC](#)
- [Backing Up the Server Parameter File](#)

Setting SPFILE Parameter Values for Oracle RAC

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

Note:

Modifying the SPFILE using tools other than Oracle Enterprise Manager or SQL*Plus can corrupt the file and prevent database startup. To repair the file, you might be required 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
```

The value before the period (.) in an SPFILE entry identifies the instance to which the particular parameter value belongs. When an asterisk (*) precedes the period, 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 Database updates the setting on all instances except the instance with SID `prod1`:

```
ALTER SYSTEM SET OPEN_CURSORS=1500 sid='' SCOPE=SPFILE;
```

The SPFILE now has the following entries for `OPEN_CURSORS`:

```
*.OPEN_CURSORS=1500
prod1.OPEN_CURSORS=1000
```

Run the following statement to reset `OPEN_CURSORS` to its default value for all instances except `prod1`:

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

The SPFILE now has just the following entry for `prod1`:

```
prod1.OPEN_CURSORS=1000
```

Run the following statement to reset the `OPEN_CURSORS` parameter to its default value for instance `prod1` only:

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

Parameter File Search Order in Oracle RAC

Oracle Database searches for your parameter file in a particular order depending on your platform. For Oracle RAC databases, you can easily determine the location of the parameter file by using the `srvctl config database` command.

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`

Note:

Oracle recommends that you do not use the default SPFILE names because all instances must use the same file and they all have different SIDs. Instead, store the SPFILE on Oracle ASM. If you store the SPFILE on a cluster file system, then use the following naming convention for the SPFILE: `$ORACLE_HOME/dbs/spfiledb_unique_name.ora`. Create a PFILE named `$ORACLE_HOME/dbs/initsid.ora` that contains the name `SPFILE=ORACLE_HOME/dbs/spfiledb_unique_name.ora`.

Backing Up the Server Parameter File

Oracle recommends that you regularly back up the server parameter file for recovery purposes.

Do this using Oracle Enterprise Manager (as described in the *Oracle Database 2 Day + Real Application Clusters Guide*) or use the `CREATE PFILE` statement. For example:

```
CREATE PFILE='/u01/oracle/dbs/test_init.ora'  
FROM SPFILE='/u01/oracle/dbs/test_spfile.ora';
```

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 does 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 Oracle Enterprise Manager or the RMAN `CONTROLFILE AUTOBACKUP` statement to enable the RMAN autobackup feature

Initialization Parameter Use in Oracle RAC

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.

Table 3-3 Initialization Parameters Specific to Oracle RAC

Parameter	Description
ACTIVE_INSTANCE_COUNT	This initialization parameter was deprecated in Oracle RAC 11g release 2 (11.2). Instead, use a service with one preferred and one available instance.
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 <code>TRUE</code> .
CLUSTER_DATABASE_INSTANCES	Oracle RAC uses this parameter to allocate adequate memory resources. It must be set to the same value on all instances. <ul style="list-style-type: none"> • For policy-managed databases, Oracle internally sets this parameter to 16 • For administrator-managed databases, Oracle internally sets it to the number of configured Oracle RAC instances <p>You can set this parameter to a value that is <i>greater than</i> the current number of instances, if you are planning to add instances. For policy-managed databases, you should set this parameter to a higher value only if you intend to run a database with more than 16 instances. In this case, set the parameter to the expected maximum number of instances on which this database will run.</p>

Table 3-3 (Cont.) Initialization Parameters Specific to Oracle RAC

Parameter	Description
CLUSTER_INTERCONNECTS	<p>Specifies an alternative cluster interconnect for the private network when there are multiple interconnects.</p> <p>Notes:</p> <ul style="list-style-type: none"> Oracle recommends that all Oracle databases and Oracle Clusterware use the same interconnect network. Oracle does not recommend setting the <code>CLUSTER_INTERCONNECTS</code> parameter except in certain situations. See "Administering Multiple Cluster Interconnects on Linux and UNIX Platforms" for more details. This parameter is stored in the Grid Plug and Play profile in a Grid Plug and Play environment.
DB_NAME	<p>If you set a value for <code>DB_NAME</code> in instance-specific parameter files, the setting must be identical for all instances.</p>
DISPATCHERS	<p>Set the <code>DISPATCHERS</code> 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 <code>DISPATCHERS</code> parameter may contain many attributes.</p> <p>Oracle recommends that you configure at least the <code>PROTOCOL</code> and <code>LISTENER</code> attributes. <code>PROTOCOL</code> specifies the network protocol for which the dispatcher process generates a listening end point. <code>LISTENER</code> 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 <code>tnsnames.ora</code> file. The <code>tnsnames.ora</code> file contains net service names. Clients, nodes, and the Oracle Performance Manager node need this file. Oracle Enterprise Manager does not require <code>tnsnames.ora</code> entries on the client for Cloud Control.</p> <p>See Also: <i>Oracle Database Net Services Administrator's Guide</i> for complete information about configuring the <code>DISPATCHERS</code> parameter and its attributes and for configuring the shared server</p>
GCS_SERVER_PROCESSES	<p>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 equals 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.</p>
INSTANCE_NAME	<p>Specifies the unique name of an instance. Clients can use this name to force their session to be connected to a specific instance in the cluster. The format of the <code>INSTANCE_NAME</code> parameter is generally <code>db_unique_name_instance_number</code>, such as <code>orclpdb_2</code>.</p> <p>Note: In Grid Plug and Play environments, the <code>INSTANCE_NAME</code> parameter is not required and defaults to <code>db_unique_name_instance_number</code> if not specified.</p>

Table 3-3 (Cont.) Initialization Parameters Specific to Oracle RAC

Parameter	Description
RESULT_CACHE_MAX_SIZE	<p>In a clustered database, you can either set <code>RESULT_CACHE_MAX_SIZE=0</code> 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:</p> <ul style="list-style-type: none"> • Enabling the result cache: Set <code>RESULT_CACHE_MAX_SIZE</code> 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 <code>RESULT_CACHE_MAX_SIZE=0</code> on all instances to disable the result cache. If you set <code>RESULT_CACHE_MAX_SIZE=0</code> upon startup of any one instance, then you must set the parameter to zero on all instance startups because disabling the result cache must be done clusterwide. Disabling the result cache on some instances may lead to incorrect results. <p>If you do not set the <code>RESULT_CACHE_MAX_SIZE</code> parameter, the parameter resolves to a default, nonzero value.</p>
SERVICE_NAMES	<p>When you use services, Oracle recommends that you do not set a value for the <code>SERVICE_NAMES</code> parameter but instead you should create cluster managed services through the Cluster Managed Services page in Oracle Enterprise Manager Cloud 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 Workload Management with Dynamic Database Services are not directly related to the features that Oracle provides when you set <code>SERVICE_NAMES</code>. In addition, setting a value for this parameter may override some benefits of using services.</p> <p>Note: Oracle recommends that client connections use services rather than instance names. Entries in the <code>SERVICE_NAMES</code> parameter may be used by client connections rather than the <code>INSTANCE_NAME</code> parameter value. The <code>SERVICE_NAMES</code> parameter may include one or more names and different instances may share one or more names with other instances, enabling 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.</p>
SPFILE	<p>When you use an SPFILE, all Oracle RAC database instances must use the SPFILE and the file must be on shared storage.</p>
THREAD	<p>Specifies the number of the redo threads to be used by an instance. You can specify any available redo thread number if 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 <code>INSTANCE_NAME</code> parameter to specify redo log groups.</p>

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:

```
COMPATIBLE
CLUSTER_DATABASE
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_EXECUTION_MESSAGE_SIZE
REMOTE_LOGIN_PASSWORDFILE
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

When it is necessary to set parameters that have unique settings on a policy-managed database, you can ensure that instances always use the same name on particular nodes by running the `srvctl modify instance -n node_name -i instance_name` command for each server that can be assigned to the database's server pool. Then a unique value of the parameter can be specified for *instance_name* that is used whenever the database runs on *node_name*.

Specify the `ORACLE_SID` environment variable, which consists of the database name and the number of the `INSTANCE_NAME` assigned to the instance.

Use the `CLUSTER_INTERCONNECTS` initialization parameter to specify an alternative interconnect to the one Oracle Clusterware is using for the private network. Each instance of the Oracle RAC database gets a unique value when setting the `CLUSTER_INTERCONNECTS` initialization parameter.

Oracle Database uses the `INSTANCE_NUMBER` parameter to distinguish among instances at startup and the `INSTANCE_NAME` parameter to assign redo log groups to specific instances. The instance name can take the form *db_unique_name_instance_number* and when it has this form of name and number separated by an underscore, the number after the underscore is used as the `INSTANCE_NUMBER`. With Oracle Database 11.2 using Grid Plug and Play, you no longer have to explicitly assign instance numbers for policy-managed databases and the instance name defaults to *db_unique_name_instance_number*, where Oracle Database assigns the instance number.

When you specify `UNDO_TABLESPACE` with automatic undo management enabled, then set this parameter to a unique undo tablespace name for each instance.

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.

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 the parameters listed here 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 either Oracle Streams downstream capture or Oracle GoldenGate integrated capture mode in a downstream capture configuration with your Oracle RAC database, the value must be greater than zero.
CLUSTER_DATABASE_INSTANCES	While it is preferable for this parameter to have identical settings across all Oracle RAC database instances, it is not required.
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 Database 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 Oracle 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 is not 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.

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

Parameter	Description
UNDO_RETENTION	By setting different values for UNDO_RETENTION in each instance, you are likely to reduce scalability and encounter unpredictable behavior following a failover. Therefore, you should carefully consider whether there are any benefits before you assign different values for this parameter to the instances in your Oracle RAC database.

Converting an Administrator-Managed Database to a Policy-Managed Database

You can convert an administrator-managed database to a policy-managed database.

Note:

If the administrator-managed database is configured for a low-privileged user and you attempt to convert the database to a policy-managed database, then you must manually add a wallet (if one does not already exist) for this low-privileged user, so that a Windows service for Oracle Database can be created.

To convert an administrator-managed database:

1. Check the current configuration of all services and the database (if you make a mistake and need to recover, then you can know what the configuration looked like when you began), as follows:

```
srvctl config database -db db_unique_name
srvctl config service -db db_unique_name
```

2. Create a server pool for the policy-managed database (you must be a cluster administrator to do this), as follows:

```
srvctl add srvpool -serverpool server_pool -min 0 -max n
```

In the preceding command, 0 is the minimum number of servers you want in the server pool and *n* is the maximum.

Note:

This step does not necessarily place servers in the newly-created server pool. If there are no servers in the Free pool from which the new server pool can allocate servers, for example, then you may have to use the `srvctl relocate server` command to relocate a server from another server pool once the conversion is complete.

3. Stop the database using Oracle Enterprise Manager or SRVCTL, as follows:

```
srvctl stop database -db db_unique_name
```

4. Modify the database to be in the new server pool, as follows:

```
srvctl modify database -db db_unique_name -serverpool server_pool
```

5. Add a service user to the wallet, as follows:

```
crsctl add wallet -type OSUSER -user user_name -passwd
```

6. Check the status of the database to confirm that it is now policy managed by repeating the commands in step 1.

Configure Oracle Enterprise Manager to recognize the change you made in the previous procedure, as follows:

1. In order for Oracle Enterprise Manager Cloud Control to recognize the new database instances, you must change the instance name from `db_unique_name#` to `db_unique_name_#` (notice the additional underscore (`_`) before the number sign (`#`) character).
2. Rename the `orapwd` file in the `dbfs/database` directory (or create a new `orapwd` file by running the `orapwd` command).

By default, there is an `orapwd` file with the instance name appended to it, such as `orapwdORCL1`. You must change the name of the file to correspond to the instance name you changed in the previous step. For example, you must change `orapwdORCL1` to `orapwdORCL_1` or create a new `orapwd` file.

You cannot directly convert a policy-managed database to an administrator-managed database. Instead, you can remove the policy-managed configuration using the `srvctl remove database` and `srvctl remove service` commands, and then register the same database as an administrator-managed database using the `srvctl add database` and `srvctl add instance` commands. Once you register the database and instance, you must use the `srvctl add service` command to add back the services as you removed them.

Services for administrator-managed databases continue to be defined by the `PREFERRED` and `AVAILABLE` definitions. For policy-managed databases, a service is defined to a database server pool and can either be *uniform* (running on all instances in the server pool) or *singleton* (running on only one instance in the server pool). If you change the management policy of the database, then you must recreate the database services to be either *uniform/singleton* or `PREFERRED/AVAILABLE`, depending upon which database management policy you choose.

Managing Memory Pressure for Database Servers

Enterprise database servers can use all available memory due to too many open sessions or runaway workloads. Running out of memory can result in failed transactions or, in extreme cases, a restart of the server and the loss of a valuable resource for your applications. Memory Guard detects [memory pressure](#) on a server in real time and redirects new sessions to other servers to prevent using all available memory on the stressed server.

When Oracle Database QoS Management is enabled and managing an Oracle Clusterware server pool, Cluster Health Monitor sends a metrics stream that provides real-time information about memory resources for the cluster servers to Memory Guard. This information includes the following:

- Amount of available memory

- Amount of memory currently in use

If Memory Guard determines that a node is experiencing memory pressure, then the database services managed by Oracle Clusterware are stopped on that node, preventing new connections from being created. After the memory stress is relieved, the services on that node are restarted automatically, and the listener starts sending new connections to that server. The memory pressure can be relieved in several ways (for example, by closing existing sessions or by user intervention).

Rerouting new sessions to different servers protects the existing workloads on the memory-stressed server and enables the server to remain available. Memory Guard is a feature of Oracle RAC that manages the memory pressure for servers, adding a new resource protection capability in managing service levels for applications hosted on Oracle RAC databases.

Quiescing Oracle RAC Databases

The procedure for quiescing Oracle RAC databases is identical to quiescing a noncluster 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. When 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 is 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 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 Database background processes may still perform updates for Oracle Database internal purposes even while the database is in quiesced state. In addition, the file headers of online data files 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.

Administering Multiple Cluster Interconnects on Linux and UNIX Platforms

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

 **Note:**

The `CLUSTER_INTERCONNECTS` initialization parameter should not be set to highly available IP (HAIP) addresses provided by Redundant Interconnect Usage. HAIP addresses are recognized automatically.

If you set multiple values for `CLUSTER_INTERCONNECTS`, then Oracle Database uses all of the network interfaces that you specify for the interconnect, providing load balancing if 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.

 **Note:**

Oracle does not recommend setting the `CLUSTER_INTERCONNECTS` initialization parameter, which overrides the default interconnect settings at the operating system level.

Instead, the best practice is to use Redundant Interconnect Usage, available with Oracle Grid Infrastructure 11g release 2 (11.2) for Oracle RAC and Oracle Real Application Clusters One Node 11g release 2 (11.2) databases, and later. For databases that precede Oracle Database 11g release 2 (11.2), use operating system-based network bonding technologies to enable high availability (and load balancing) for network interface cards meant to be used as the cluster interconnect. If you want to use multiple database versions in one cluster, you can combine both techniques. Redundant Interconnect Usage will use the interfaces as presented on the operating system level, regardless of bonding. For more information regarding bonding technologies contact your operating system vendor.

Use Cases for Setting the `CLUSTER_INTERCONNECTS` Parameter

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.

 **Note:**

- Oracle does not recommend setting the `CLUSTER_INTERCONNECTS` parameter when using a policy-managed database.
- Oracle recommends that all databases and Oracle Clusterware use the same interconnect network.

Typically, you set the `CLUSTER_INTERCONNECTS` parameter only in the following situations:

- The cluster is running multiple databases and you need the interconnect traffic to be separated and you do not use Redundant Interconnect Usage.
- 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 want to use Redundant Interconnect Usage.
- 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.
- 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 the first instance on `node1` lists the IP addresses of the `alt0:`, `fta0:`, and `ics0:` devices in that order, then the parameter for the second instance on `node2` must list the IP addresses of the equivalent network adapters in the same order.
- If an operating system error occurs while Oracle Database is writing to the interconnect that you specify with the `CLUSTER_INTERCONNECTS` parameter, then Oracle Database 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.

Example

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 that cannot use Redundant Interconnect Usage.

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 `ipn` is an IP address in standard dot-decimal format, for example: 144.25.16.214:

```
Database One: crml.CLUSTER_INTERCONNECTS = ip1  
Database Two: ext1.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
```

Customizing How Oracle Clusterware Manages Oracle RAC Databases

Use these examples to minimize Oracle Clusterware control over Oracle RAC databases, which you may need to do during upgrades.

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, for example, during database upgrades.

Note:

When using third-party clusterware, Oracle recommends that you use Oracle Clusterware to manage the Oracle RAC instances. If you set the instance to manual and start it with third-party clusterware, then do not use the third-party clusterware to monitor and restart database instances because 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 management policy to define the degree of control. There are two management policies: `AUTOMATIC`, which is the default, and `MANUAL`. If the management policy is set to `AUTOMATIC`, the database is automatically restored to its previous running condition (started or stopped) upon restart of the database host computer. If `MANUAL`, the database is never automatically restarted upon restart of the database host computer. A `MANUAL` setting does not prevent Oracle Restart from monitoring the database while it is running and restarting it if a failure occurs.

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

Example 1: Display the Current Management Policy

Use the following command syntax to display the current management policy where *db_unique_name* is the name of the database for which you want to change management policies:

```
srvctl config database -db db_unique_name -all
```

Example 2: Change the Current Management Policy to Another Management Policy

Use the following SRVCTL command syntax to change the current management policy to either AUTOMATIC, MANUAL, or NORESTART:

```
srvctl modify database -db db_unique_name -policy [AUTOMATIC | MANUAL | NORESTART]
```

This command syntax sets the resource attribute of the database resource.

Example 3: Specify a Management Policy for a New Database

When you add a new database using the `srvctl add database` command, you can use the `-policy` parameter to specify the management policy as either AUTOMATIC, MANUAL, or NORESTART, as shown in the following example where *db_unique_name* is the name of the database:

```
srvctl add database -db db_unique_name -policy [AUTOMATIC | MANUAL | NORESTART]
-oraclehome $ORACLE_HOME -dbname DATA
```

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

Advanced Oracle Enterprise Manager Administration

You can install, configure, and monitor an Oracle RAC database from a single location using Oracle Enterprise Manager Cloud 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 RAC Databases".

This section includes the following topics:

- [Using Oracle Enterprise Manager Cloud Control to Discover Nodes and Instances](#)
- [Other Oracle Enterprise Manager Capabilities](#)
- [Administering Jobs and Alerts in Oracle RAC](#)

Using Oracle Enterprise Manager Cloud Control to Discover Nodes and Instances

Discovering Oracle RAC database and instance targets in Oracle Enterprise Manager enables monitoring and administration.

Oracle Enterprise Manager Cloud Control enables you to use the Oracle Enterprise Manager console interface to discover Oracle RAC database and instance targets.

If the Oracle Enterprise Manager Cloud Control agents are installed on a cluster that has an Oracle RAC database, then 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 Oracle Enterprise Manager Cloud Control as follows:

1. Log in to Oracle 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 Oracle 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 Oracle Enterprise Manager to discover only noncluster databases and listeners, or to discover all cluster databases, noncluster databases, and listeners on the cluster, then click **Continue**.
6. If this procedure did not discover your reconfigured cluster database and all of its instances, you can use the Targets Discovered on Cluster page to manually configure your cluster database and noncluster databases.

Other Oracle Enterprise Manager Capabilities

This section lists Oracle Enterprise Manager capabilities available with Oracle Enterprise Manager 12c.

- The Oracle Grid Infrastructure/Oracle RAC Provisioning deployment procedure provisions Oracle RAC 12c and Oracle Grid Infrastructure. This procedure also has a feature called *Profiles*, which enables you to record the inputs and subsequently use them for repeated deployments.
- Dynamic prerequisites for the new procedures enable Oracle Enterprise Manager, when connected to My Oracle Support (formerly *OracleMetaLink*), to download the latest prerequisites and tools for Oracle RAC provisioning.
- The existing *One-Click Extend Cluster Database* capability now supports Oracle RAC 12c stack.
- The existing *Delete/Scale down Oracle Real Application Clusters* capability is certified with Oracle RAC 12c clusters.
- The existing *Oracle Database Provisioning* procedure now supports provisioning of single instances of Oracle Database 12c.
- A new deployment procedure—Oracle Grid Infrastructure Provisioning for Standalone Servers—has been introduced to provision Oracle Grid Infrastructure 12c for noncluster databases.

Administering Jobs and Alerts in Oracle RAC

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.

This section includes the following topics:

- [Administering Jobs in Oracle RAC](#)
- [Administering Alerts in Oracle RAC with Oracle Enterprise Manager](#)
- [Using Defined Blackouts in Oracle Enterprise Manager](#)

Administering Jobs in Oracle RAC

You can administer Oracle Enterprise Manager jobs at both the database and instance levels. For example, you can create a job at the cluster database level to run on any active instance of the target Oracle RAC database. Or you can create a job at the instance level to run on the specific instance for which you created it. If there is a failure, then 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. Oracle 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 Oracle Enterprise Manager displays the Database Home page, click **Maintenance**.
4. Complete the Enterprise Manage Job Wizard pages to create the job.

Administering Alerts in Oracle RAC with Oracle Enterprise Manager

You can use Oracle 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.

Oracle 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 function 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, to avoid receiving duplicate alerts at each instance.

Using Defined Blackouts in Oracle Enterprise Manager

You can define blackouts (which are time periods in which database monitoring is suspended so that maintenance operations do not skew monitoring data or generate needless alerts) 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

Administering Oracle RAC One Node

Oracle Real Application Clusters One Node (Oracle RAC One Node) is a single instance of an Oracle Real Application Clusters (Oracle RAC) database that runs on one node in a cluster. This option adds to the flexibility that Oracle offers for database consolidation. You can consolidate many databases into one cluster with minimal overhead while also providing the high availability benefits of failover protection, online rolling patch application, and rolling upgrades for the operating system and Oracle Clusterware.

This chapter includes the following topics:

- [Creating an Oracle RAC One Node Database](#)
- [Converting Databases](#)
- [Online Database Relocation](#)

Creating an Oracle RAC One Node Database

You can create Oracle RAC One Node databases by using Rapid Home Provisioning or the Database Configuration Assistant (DBCA), as with any other Oracle database (manually created scripts are also a valid alternative).

You can create an Oracle RAC One Node database using Rapid Home Provisioning and the `rhptl add database` command with the `-dbtype RACONENODE` parameter. You can also include an Oracle RAC One Node database using the `rhptl add workingcopy` command.

Oracle RAC One Node databases may also be the result of a conversion from either a single-instance Oracle database (using `rconfig`, for example) or an Oracle RAC database. Typically, Oracle-provided tools register the Oracle RAC One Node database with Oracle Clusterware. Depending on your configuration, automatic registration of an Oracle RAC One Node database with Oracle Clusterware may not have happened. If this is the case, then follow the steps in this section to register the Oracle RAC One Node database with Oracle Clusterware.

Note:

Oracle recommends that you manage Oracle RAC One Node databases with Server Control Utility (SRVCTL). You can only perform certain operations (such as Online Database Relocation) using SRVCTL.

If your Oracle RAC One Node database did not register automatically with Oracle Clusterware, then use the `srvctl add database` command to add an Oracle RAC One Node database to your cluster. For example:

```
$ srvctl add database -dbtype RACONENODE [-server server_list]
  [-instance instance_name] [-timeout timeout]
```

Use the `-server` option and the `-instance` option when adding an administrator-managed Oracle RAC One Node database.

For Oracle RAC One Node databases, you must configure at least one dynamic database service (in addition to and opposed to the default database service). When using an administrator-managed Oracle RAC One Node database, service registration is performed as with any other Oracle RAC database. When you add services to a policy-managed Oracle RAC One Node database, SRVCTL does not accept any placement information, but instead configures those services using the value of the `SERVER_POOLS` attribute.

 **Note:**

When adding an administrator-managed Oracle RAC One Node database, you can optionally supply an instance prefix with the `-instance instance_name` option of the `srvctl add database` command. The name of the instance will then be `prefix_1`. If you do not specify an instance prefix, then the first 12 characters of the unique name of the database becomes the prefix. The instance name changes to `prefix_2` during an online database relocation and reverts back to `prefix_1` during a subsequent online database relocation. The same instance name is used on failover.

Converting Databases

Using SRVCTL, you can convert an Oracle RAC database with one instance to an Oracle RAC One Node database and vice versa.

This section includes the following topics:

- [Converting a Database from Oracle RAC to Oracle RAC One Node](#)
- [Converting a Database from Oracle RAC One Node to Oracle RAC](#)

Converting a Database from Oracle RAC to Oracle RAC One Node

Use SRVCTL to convert an Oracle RAC database to an Oracle RAC One Node database.

Before converting an Oracle RAC database to an Oracle RAC One Node database, you must first ensure that the Oracle RAC database has only one instance. If your Oracle RAC database is administrator managed and has more than one instance, then you must remove all instances except one using the `srvctl remove instance` command. If your Oracle RAC database is policy managed and has more than one instance, then you must stop all instances except one using the `srvctl stop instance` command.

If the Oracle RAC database is administrator managed, then you must change the configuration of all services to set the preferred instance to the instance that you want to keep as an Oracle RAC One Node database after conversion. If any service had a `PRECONNECT TAF` policy, then its TAF policy must be updated to `BASIC` or `NONE` before starting the conversion process. These services must no longer have any available instance.

If the Oracle RAC database is policy managed, then you must change the configuration of all services so that they all use the same server pool before you convert the Oracle RAC database to an Oracle RAC One Node database.

You can convert an Oracle RAC database with one instance to an Oracle RAC One Node database using the `srvctl convert database` command, as follows:

```
$ srvctl convert database -db db_unique_name -dbtype RACONENODE  
[-instance instance_name -timeout timeout]  
-w timeout]
```

 **Note:**

An Oracle RAC database that you want to convert to Oracle RAC One Node must either use Oracle Managed Files (to enable automatic thread allocation) or have at least two redo threads.

Converting a Database from Oracle RAC One Node to Oracle RAC

You can convert an Oracle RAC One Node database to an Oracle RAC database by logging in as the Oracle RAC One Node database owner and entering the following SRVCTL command:

```
srvctl convert database -db db_unique_name -dbtype RAC
```

If you are relocating the database you want to convert to Oracle RAC using online database relocation, or an online database relocation has failed, then you must either quit or complete the relocation before you run the `srvctl convert database` command.

After you run this command, you must create server pools for each database service, in addition to the database server pool. The values for `SERVER_NAMES` for the server pools used by the database services must be set to the node that you converted from an Oracle RAC One Node to an Oracle RAC node. You can use the CRSD utility or Oracle Enterprise Manager to create and configure the server pools.

Converting an administrator-managed Oracle RAC One Node database to an Oracle RAC database configures all database services so that the single-instance database is the preferred instance for that service. After you convert the database, you can add instances to your database by using the `srvctl add instance` command.

Converting a policy-managed Oracle RAC One Node database to an Oracle RAC database sets all database services to UNIFORM cardinality. It also results in reusing the server pool in which the database currently runs. The conversion reconfigures the database to run on all nodes in the server pool. The command does not start any additional instances but running the `srvctl start database` command starts the database on all nodes in the server pool.

Online Database Relocation

You can relocate an Oracle RAC One Node database to another node while still maintaining service availability using the online database relocation feature.

Only during a planned online database relocation is a second instance of an Oracle RAC One Node database created, so that any database sessions can continue while

the database is relocated to a new node. You can only use online database relocation with Oracle RAC One Node databases but you cannot use online database relocation with Oracle RAC databases regardless of their management style (either administrator or policy managed).

You can use the `srvctl relocate database` command to configure the amount of time after the relocated database starts and services are migrated, before the former instance of the database stops. This configured amount of time is not an upper bound on the amount of time taken by the entire operation, but only controls how long the relocated database waits for connections to migrate from the former instance to the new instance, before stopping the former instance.

Online database relocation occurs, as follows:

1. Start a new database instance in a different location.
2. Move all the services to the relocated instance.
3. Wait for all the connections to migrate to the relocated instance.
4. Shut down the former database instance, forcing any remaining connections to move to the relocated instance.

The online relocation timeout is the amount of time you configure to complete step 3.

If your Oracle RAC One Node database is administrator managed, then the target node to which you want to relocate the database instance must be part of the Free server pool at the moment you initiate the relocation. If you have a spare target node in the Free server pool, then this node is added to the candidate list for the Oracle RAC One Node database.

 **Note:**

- When you relocate a database instance to a target node that is not currently in the candidate server list for the database, you must copy the password file, if configured, to the target node, unless you use shared password files stored in Oracle ASM.

When you use password file-based authentication for remote management of Oracle RAC One Node databases without any shared password file, you must have two password files on each node where the database can run: one named `SID_prefix_1` and the other named `SID_prefix_2`. You must recopy both of these files to all candidate nodes every time you update the password file. This is true for both policy-managed and administrator-managed databases.

Oracle recommends using Oracle Clusterware to start and stop the database, and defining users in the data dictionary for other management.

- Before you relocate a database instance, you must ensure that the database service user is added to the wallet. Run `crsctl query wallet -type OSUSER -all` to check whether the database service user is in the wallet. If not, then run `crsctl add wallet -type OSUSER -user user_name -passwd` to add the database service user to the wallet.

Use the `srvctl relocate database` command to initiate relocation of an Oracle RAC One Node database. For example:


```
$ srvctl relocate database -d rac1 -n node7
```

5

Workload Management with Dynamic Database Services

Workload management includes load balancing, enabling clients for Oracle Real Application Clusters (Oracle RAC), distributed transaction processing, and services.

This chapter includes the following topics:

- [Connection Load Balancing](#)
- [Load Balancing Advisory](#)
- [Enabling Clients for Oracle RAC](#)
- [Distributed Transaction Processing in Oracle RAC](#)
- [Automatic Workload Repository](#)
- [Measuring Performance by Service Using the Automatic Workload Repository](#)
- [Automatic Workload Repository Service Thresholds and Alerts](#)
- [Using Oracle Services](#)
- [Service Deployment Options](#)
- [Administering Services](#)
- [Global Services](#)
- [Connecting to a Service: An Example](#)

Connection Load Balancing

Oracle Net Services provides the ability to distribute 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 distributes the connection requests across the listeners, independently at each client. With server-side load balancing, the SCAN listener directs a connection request to the best instance currently providing the service, based on the `-clbgoal` and `-rlbgoal` settings for the service.

The SCAN listener is aware of the HTTP protocol so that it can redirect HTTP clients to the appropriate handler, which can reside on different nodes in the cluster than the node on which the SCAN listener resides.

In an Oracle RAC database, client connections should use both types of connection load balancing.

- [Server-Side Load Balancing](#)
- [Generic Database Clients](#)
- [Client-Side Load Balancing](#)
- [Client-Side Connection Configuration for Older Clients](#)

Server-Side Load Balancing

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

- Configures and enables server-side load balancing
- Creates a sample client-side load balancing connection definition in the `tnsnames.ora` file on the server

The Oracle Clusterware Database Agent is responsible for managing the `LISTENER_NETWORKS` parameter.



Note:

Note: If you set the `REMOTE_LISTENER` parameter manually, then set this parameter to `scan_name:scan_port`.

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:

- **SHORT:** Use the `SHORT` connection load balancing method for applications that use run-time load balancing. When using connection pools that are integrated with Load Balancing Advisory, set the `CLB_GOAL` to `SHORT`. The following example modifies the service known as `oltpapp`, using `SRVCTL` to set the connection load balancing goal to `SHORT`:

```
$ srvctl modify service -db db_unique_name -service oltpapp -clbgoal SHORT
```

- **LONG:** Use the `LONG` connection load balancing method if run-time load balancing is not required. This is typical for batch operations. `LONG` is the default connection load balancing goal. The following is an example of modifying a service, `batchconn`, using `SRVCTL` to define the connection load balancing goal for long-lived sessions:

```
$ srvctl modify service -db db_unique_name -service batchconn -clbgoal LONG
```

Generic Database Clients

Oracle Net Services enables you to add the `CONNECT_TIMEOUT`, `RETRY_COUNT`, and `TRANSPORT_CONNECT_TIMEOUT` parameters to the `tnsnames.ora` connection string.

For example, when using `SCAN` addresses for the remote listeners at the database:

```
jdbc:oracle:thin:@(DESCRIPTION =
  (TRANSPORT_CONNECT_TIMEOUT=3)(CONNECT_TIMEOUT=60)
  (RETRY_COUNT=3)(FAILOVER=ON)
  (ADDRESS_LIST =(ADDRESS=(PROTOCOL=tcp)
    (HOST=CLOUD-SCANVIP.example.com)(PORT=5221))
  (CONNECT_DATA=(SERVICE_NAME=orcl)))
  Remote_listeners=CLOUD-SCANVIP.example.com:5221
```

For example, when using remote listeners pointing to VIPs at the database:

```

jdbc:oracle:thin:@(DESCRIPTION =
  (TRANSPORT_CONNECT_TIMEOUT=3)
  (CONNECT_TIMEOUT=60) (RETRY_COUNT=20)
  (RETRY_DELAY=3) (FAILOVER=ON)
  (ADDRESS_LIST=
    (ADDRESS=(PROTOCOL=tcp)(HOST=CLOUD-VIP1)(PORT=1521) )
    (ADDRESS=(PROTOCOL=tcp)(HOST=CLOUD-VIP2)(PORT=1521) )
    (ADDRESS=(PROTOCOL=tcp)(HOST=CLOUD-VIP3)(PORT=1521) ))
  (CONNECT_DATA=(SERVICE_NAME=GOLD)))

```

The value of these parameters is expressed in seconds. In the preceding examples, Oracle Net waits for 60 seconds for each full connection to receive a response, after which it assumes that a failure occurred and retries the next address in the ADDRESS_LIST. Oracle Net retries the address list 3 times before it returns a failure message to the client. The TRANSPORT_CONNECT_TIMEOUT parameter establishes the time to wait to establish a TCP connection to the database server.

For SCAN, Oracle Net Services tries all three addresses (returned by the SCAN address) before returning a failure to the client. EZConnect with SCAN includes this connection failover feature.

This behavior is called Oracle Net *connection failover*. If an error is returned from a chosen address in the list, then Oracle Net Services tries the next address in the list until it is either successful or it has exhausted all addresses in its list.

Client-Side Connection Configuration for Older Clients

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 tries the next address in the list until it is either successful or it has exhausted all addresses in its list. For SCAN, Oracle Net Services tries all three addresses before returning a failure to the client. EZConnect with SCAN includes this connection failover feature.

To increase availability, you can specify a timeout that specifies how long Oracle Net waits for a response from the listener before returning an error. The method of setting this timeout parameter depends on the type of client access. Oracle Net maintains these parameters for backward compatibility.

This section includes the following topics:

- [JDBC-Thin Clients](#)
- [OCI Clients](#)

JDBC-Thin Clients

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.

OCI Clients

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 = number_of_seconds
```

The granularity of the timeout value for the OCI client is in seconds. The `sqlnet.ora` file affects all connections using this client.



Note:

Do *not* configure the connection timeout in the `sqlnet.ora` file on the server.

Client-Side Load Balancing

Client-side load balancing is defined in your client connection definition (`tnsnames.ora` file, for example) by setting the parameter `LOAD_BALANCE=ON`. When you set this parameter to `ON`, Oracle Database randomly selects an address in the address list, and connects to that node's listener. This balances client connections across the available SCAN listeners in the cluster.

If you configured SCAN for connection requests, then client-side load balancing is not relevant for those clients that support SCAN access. When clients connect using SCAN, Oracle Net automatically balances the load of client connection requests across the three IP addresses you defined for the SCAN, unless you are using EZConnect.

The SCAN listener redirects the connection request to the local listener of the instance that is least loaded (if `-clbgoal` is set to `SHORT`) and provides the requested service. 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.

When clients connect using SCAN, Oracle Net automatically load balances client connection requests across the three IP addresses you defined for the SCAN, unless you are using EZConnect.

If you are using clients that do not support SCAN (for example, the client version is pre-Oracle Database 11g release 2 (11.2)), then, to use SCAN you must change the client `tnsnames.ora` to include the SCAN VIPs, and set `LOAD_BALANCE=ON` to balance requests across the VIPs. For example:

```
Sales.example.com=(DESCRIPTION=
  (ADDRESS_LIST=(LOAD_BALANCE=ON)(FAILOVER=ON)
    (ADDRESS=(PROTOCOL=TCP)(HOST=172.22.67.192)(PORT=1521))
    (ADDRESS=(PROTOCOL=TCP)(HOST=172.22.67.193)(PORT=1521))
    (ADDRESS=(PROTOCOL=TCP)(HOST=172.22.67.194)(PORT=1521))
    (CONNECT_DATA=(SERVICE_NAME=salesservice.example.com))
  ))
```

 **Note:**

If your database is not Oracle Database 11g release 2 (11.2), or later, and you want to use SCAN, then you must add SCAN VIPs to the `REMOTE_LISTENER` parameter to enable correct listener cross-registration.

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](#)
- [Monitoring 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 connection pools with persistent connections that span the instances that offer a particular service. When using persistent connections, 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 Load Balancing Advisory and run-time connection load balancing 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 universal connection pool, OCI session pool, Oracle WebLogic Server Active GridLink for Oracle RAC, and the ODP.NET Connection Pools. Third-party applications can also subscribe to load balancing advisory events by using JDBC and Oracle RAC FAN API or by using callbacks with OCI.

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.

Configuring a service-level goal enables the load balancing advisory and the publishing of FAN load balancing events for that service. There are two types of service-level goals for run-time connection load balancing:

- **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. The following example shows how to set the goal to **SERVICE_TIME** for connections using the `online` service:

```
$ srvctl modify service -db db_unique_name -service online
  -rlbgoal SERVICE_TIME -clbgoal 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. The following example shows how to set the goal to **THROUGHPUT** for connections using the `sjob` service:

```
$ srvctl modify service -db db_unique_name -service sjob
  -rlbgoal THROUGHPUT -clbgoal LONG
```

Setting the run-time connection load balancing goal to `NONE` disables load balancing for the service. You can see the goal settings for a service in the data dictionary by querying the `DBA_SERVICES`, `V$SERVICES`, and `V$ACTIVE_SERVICES` views. You can also review the load balancing settings for a service using Oracle Enterprise Manager.

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 run-time connection load balancing feature of an Oracle integrated client such as JDBC, Universal Connection Pool (or the deprecated Implicit Connection Cache), ODP.NET Connection Pools, OCI session pools, or Oracle WebLogic Server Active GridLink for Oracle RAC. Other client applications can take advantage of FAN programmatically by using the Oracle RAC FAN API to subscribe to FAN events and execute event-handling actions upon receipt. [Table 5-1](#) describes the load balancing advisory FAN event parameters.

See Also:

Oracle Database JDBC Developer's Guide for more information about the Oracle RAC FAN API

Table 5-1 Load Balancing Advisory FAN Events

Parameter	Description
VERSION	Version of the event record. Used to identify release changes.
EVENT_TYPE	A load balancing advisory event is always of the <code>SERVICEMETRICS</code> event type.

Table 5-1 (Cont.) Load Balancing Advisory FAN Events

Parameter	Description
SERVICE	The service name; matches the value of NAME in DBA_SERVICES.
DATABASE	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.

 **Note:**

The INSTANCE, PERCENT, and FLAG event parameters are generated for each instance offering the service. Each set of instance data is enclosed within braces ({}).

Monitoring Load Balancing Advisory FAN Events

You can use the following query against the internal queue table for load balancing advisory FAN events to monitor load balancing advisory events generated for an instance:

```
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 ;
```

The results of this query contain rows similar to the following:

```
02:56:05|SYS$RLBTYP('hr', 'VERSION=1.0 database=sales service=hr
  { {instance=sales_4 percent=38 flag=GOOD aff=TRUE}{instance=sales_1
    percent=62 flag=GOOD aff=TRUE} } timestamp=2012-07-16 07:56:05')
```

Following is an example of a load balancing advisory event for the lba_serv service offered on two instances (orc11 and orc12), as captured from Oracle Notification Service using the Oracle RAC FAN API:

```
Notification Type: database/event/servicemetrics/lba_serv.example.com
VERSION=1.0 database=orcl service=lba_serv.example.com { {instance=orc12
  percent=50 flag=UNKNOWN aff=FALSE}{instance=orc11 percent=50 flag=UNKNOWN
  aff=FALSE} } timestamp=2012-07-06 13:19:12
```




Note:

The `SERVICEMETRICS` events are not visible through the FAN callout mechanism.

Enabling Clients for Oracle RAC

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.

The following sections discuss how FAN is integrated with Oracle Clients and how to enable FAN events for the several specific client development environments:

- [Overview of Oracle Integrated Clients and FAN](#)
- [Enabling JDBC-Thin Clients for Fast Connection Failover](#)
- [Enabling JDBC Clients for Run-time Connection Load Balancing](#)
- [Configuring JDBC-Thin Clients for Application Continuity for Java](#)
- [Configuring JDBC-Thin Clients for Transaction Guard](#)
- [Enabling OCI Clients for Fast Connection Failover](#)
- [Enabling OCI Clients for Run-time Connection Load Balancing](#)
- [Configuring OCI Clients to use Transaction Guard](#)
- [Enabling ODP.NET Clients to Receive FAN High Availability Events](#)
- [Enabling ODP.NET Clients to Receive FAN Load Balancing Advisory Events](#)
- [Configuring ODP.NET Clients to use Transaction Guard](#)

Overview of Oracle Integrated Clients and FAN

The overall goals of FAN are to enable end-to-end, lights-out recovery of applications and load balancing based on real transaction performance.

Applications use the FAN high availability (HA) events to achieve very fast detection of failures, balancing of connection pools following failures, and distribution of connections again when the failed components are repaired.

The FAN events carrying load balancing advice help connection pools consistently deliver connections to available instances that provide the best service. FAN HA is integrated with the JDBC-thin and OCI drivers. FAN HA and FAN load balancing are both integrated with the JDBC Universal Connection Pool (and the deprecated Implicit Connection Cache), the OCI session pools, the ODP.NET connection pool, and Oracle WebLogic Server Active GridLink for Oracle RAC.

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 instances or services that are 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.

Oracle client drivers that are integrated with FAN can:

- 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 Database detects the `NOT RESTARTING` state, instead of making the client wait while the service repeatedly attempts to restart

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
- Balance work requests at run time using load balancing advisory events

The use of client drivers or connection pools and FAN requires that you properly configure the Oracle Notification Service to deliver the FAN events to the clients. In addition, for load balancing, configure database connection load balancing across all of the instances that provide the services used by the connection pool. Oracle recommends that you configure both client-side and server-side load balancing with Oracle Net Services. If you use DBCA to create your database, then both client-side and server-side load balancing are configured by default.

Enabling JDBC-Thin Clients for Fast Connection Failover

Enabling Fast Connection Failover (FCF) for Universal Connection Pool and Oracle WebLogic Server Active GridLink for Oracle RAC enables the use of FAN HA and load balancing advisory events.

For Universal Connection Pool to use FAN, your application can use the JDBC development environment for either JDBC OCI or JDBC Thin clients. The Java Database Connectivity Oracle Call Interface (JDBC/OCI) driver connection pooling functionality is part of the JDBC-thin client. This functionality is provided by the `OracleOCIConnectionPool` class.

To enable FCF for the JDBC-thin client, call the method `setFastConnectionFailoverEnabled(true)` of the `OracleDataSource` class in the `oracle.jdbc.pool` package before making the first `getConnection()` request. When you enable FCF for the JDBC-thin client, the failover property applies to every connection in the connection pool. Enabling FCF with JDBC-thin driver or JDBC/OCI clients enables the connection pool to receive and react to all FAN events.

JDBC application developers can programmatically integrate with FAN by using a set of APIs introduced in Oracle Database 11g release 2 (11.2). The Oracle RAC FAN APIs enable application code to receive and respond to FAN event notifications sent by Oracle RAC in the following ways:

- Listening for Oracle RAC service down, service up, and node down events
- Listening for load balancing advisory events and responding to them

Oracle Notification Service for JDBC-Thin Clients

FCF relies on Oracle Notification Service to propagate database events between the connection pool and the Oracle RAC database. At run time, the connection pool must be able to setup an Oracle Notification Service environment. Oracle Notification Service (`ons.jar`) is included as part of the Oracle Client software. Oracle Notification

Service can be configured using either remote configuration or client-side Oracle Notification Service daemon configuration. Remote Oracle Notification Service subscription offers the following advantages:

- Support for an All Java mid-tier software
- An Oracle Notification Service daemon is not necessary on the client system, so you do not have to manage this process
- Simple configuration by way of a `DataSource` property

Configuring FCF for JDBC/OCI and JDBC-Thin Driver Clients

You can enable FCF for Universal Connection Pool or Implicit Connection Cache.

Oracle recommends using the Universal Connection Pool for Java because the Implicit Connection Cache is deprecated. You can also use Oracle WebLogic Server Active GridLink for Oracle RAC.

This procedure explains how to enable FCF for JDBC. For JDBC/OCI clients, if you enable FCF, then do not use the method used with Oracle Database 11g release 2 (11.2) of enabling FAN for OCI clients (setting `notification` to `TRUE` on the service), and do not configure TAF, either on the client or for the service. You can also configure Application Continuity and Transaction Guard.

To enable FCF, you must first enable the Universal Connection Pool, as described in the following procedure:

1. Create the connection pool and set `setFastConnectionFailoverEnabled(true)`.

The following example creates a connection pool and enables FCF. The `ucp.jar` library must be included in the classpath of an application to use this example.

```
PoolDataSource pds = PoolDataSourceFactory.getPoolDataSource();  
pds.setFastConnectionFailoverEnabled(true);
```

2. Determine the ports to use for Oracle Notification Service remote subscriptions.

Use the following command to view the Oracle Notification Service configuration on each node that is running Oracle Clusterware as in the following example:

```
srvctl config nodeapps -ononly
```

The output of this command lists the local and remote ports configured for Oracle Notification Service.

Note:

Oracle Notification Service configuration should have been automatically completed during the Oracle Clusterware installation.

3. Configure the remote Oracle Notification Service subscription.

When using the Universal Connection Pool, an application calls `setONSConfiguration` for an `OracleDataSource` instance and specifies the nodes and port numbers to use. The port numbers used for each node are the same as the remote port displayed for each node in Step 2, as shown in the following example. The `ons.jar` library must be included in the classpath of an application to use this example.

```
pds.setONSConfiguration("nodes=racnode1:6200, racnode2:6200");
```

Applications that use remote Oracle Notification Service configuration must set the `oracle.ons.oraclehome` system property to the location of `ORACLE_HOME` before starting the application, for example:

```
java -Doracle.ons.oraclehome=$ORACLE_HOME ...
```

4. Configure the connection URL.

A connection factory's connection URL must use the service name syntax when using FCF. The service name is used to map the connection pool to the service. The following example demonstrates configuring the connection URL:

```
pds.setConnectionFactoryClassName("oracle.jdbc.pool.OracleDataSource");
pds.setURL("jdbc:oracle:thin://SCAN_name:service_name");...
```

Enabling JDBC Clients for Run-time Connection Load Balancing

Run-time connection load balancing requires the use of an Oracle JDBC driver and an Oracle RAC database.

Oracle JDBC Universal Connection Pool and Oracle WebLogic Server Active GridLink for Oracle RAC leverage the load balancing functionality provided by an Oracle RAC database.

The Universal Connection Pool and Oracle WebLogic Server Active GridLink for Oracle RAC are integrated to take advantage of Load Balancing Advisory information. Oracle introduced the Universal Connection Pool for JDBC in Oracle Database 11g release 11.1.0.7.0. Consequently, Oracle deprecated the existing JDBC connection pool, the Implicit Connection Cache, which was introduced in Oracle Database 10g release 1 for use with Oracle RAC databases. In addition to Oracle Database 12c, you can also use the Universal Connection Pool with Oracle Database 10g or Oracle Database 11g.

Run-time connection load balancing requires that FCF is enabled and configured properly. In addition, the Oracle RAC load balancing advisory must be configured with service-level goals for each service used by the connection pool. The connection load balancing goal should be set to `SHORT`, for example:

```
srvctl modify service -db db_unique_name -service service_name
-rlbgoal SERVICE_TIME -clbgoal SHORT
```

Configuring JDBC-Thin Clients for Application Continuity for Java

The Replay data source (`oracle.jdbc.replay.OracleDataSource`) is a JDBC-thin data source that Application Continuity requires for Java.

This data source serves as the connection factory that produces new physical JDBC connections, for both Universal Connection Pool and Oracle WebLogic Server Active GridLink for Oracle RAC data sources. The JDBC replay driver maintains a history of calls during a client conversation with Oracle Database 12c, in collaboration with Oracle Database. Following any outage of the session caused by a loss of database service, planned or unplanned, under the direction of the database, the JDBC replay driver attempts to rebuild the non-transactional and transactional database session states, so that the outage appears as a delayed execution.

To use Application Continuity for Java and the JDBC replay driver, you must use an Oracle Database 12c client and connect to an Oracle Database 12c database. Application Continuity for Java is supported in the following configurations:

- JDBC applications using Oracle JDBC Replay data source and using neither Universal Connection Pool or Oracle WebLogic Server Active GridLink—typical third-party, JDBC-based connection pools
- JDBC applications using Universal Connection Pool data sources—standalone or third-party application servers configured to use a Universal Connection Pool data source
- JDBC applications using only Oracle WebLogic Server Active GridLink but not Universal Connection Pool data sources—typical Oracle WebLogic Server J2EE cases

To configure JDBC-thin clients to use the JDBC Replay Driver:

1. Ensure that you are using an application that is certified for replay.
2. Use SRVCTL to create a service for use by the application, if one does not already exist. Set the `-failovertype` parameter to `TRANSACTION` and the `-commit_outcome` parameter to `TRUE` for this service.
3. Configure the connection element using the `replayDataSource` object, as shown in the following example:

```
replayDataSource rds = PoolDataSourceFactory.getreplayDataSource();
rds.setConnectionPoolName("replayExample");
rds.setONSConfiguration("nodes=racnode1:4200, racnode2:4200");
rds.setFastConnectionFailoverEnabled(true);
rds.setConnectionFactoryClassName("oracle.jdbc.replay.OracleDataSource");
```

```
Connection conn = replayDataSource.getConnection();
```

4. When connecting to the database, use a URL that can access all instances offering the service.

Configuring JDBC-Thin Clients for Transaction Guard

Transaction Guard provides a protocol and a generic tool for applications to use for at-most-once execution in case of planned and unplanned outages.

Applications use the logical transaction ID to determine the outcome of the last transaction open in a database session following an outage. Without Transaction Guard, end users or applications that attempt to retry operations following outages can cause logical corruption by committing duplicate transactions or committing transactions out of order.

Enabling OCI Clients for Fast Connection Failover

OCI clients can enable FCF by registering to receive notifications for Oracle RAC high availability FAN events and responding when events occur. Using FCF improves the session failover response time in OCI applications and also removes connections to nonfunctioning instances from connection and session pools. FCF can be used in OCI applications that also use TAF, OCI drivers (including your own connection pools), OCI connection pool, and OCI session pools. FAN is posted over the Oracle Notification Service for both high availability and load balancing events.

To use FCF, you must use a service with FAN enabled. FAN is published over Oracle Notification Service. Client applications can also 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 if the instance it is connected to fails.

If your application is using TAF, then you must enable the TAF properties for the service using `SRVCTL` or Oracle Enterprise Manager. Configure your OCI client applications to connect to an Oracle RAC database using the configured service.

 **Note:**

OCI does not manage `UP` events.

Configuring FCF for OCI Clients

OCI applications must connect to an Oracle RAC instance to enable HA event notification. Furthermore, these applications must perform the following steps to configure FCF for an OCI client:

1. Configure the service for your OCI connection pool to enable FAN, connection load balancing, and run-time connection load balancing, as shown in the following example:

```
$ srvctl modify service -db crm -service ociapp.example.com -notification TRUE
```

2. Link the application with a thread library.
3. After linking with the thread library, the applications can register a callback that is invoked whenever a FAN event occurs.

Enabling OCI Clients for Run-time Connection Load Balancing

As of Oracle Database 12c, OCI session pooling enables multiple threads of an application to use a dynamically managed set of pre-created database sessions.

In connection pooling, the pool element is a connection, but in session pooling, the pool element is a session. Oracle Database continually reuses the sessions in the session pool to form nearly permanent channels to the instances, thus saving the overhead of creating and closing sessions every time applications need them.

Run-time connection load balancing is enabled by default in an Oracle Database 11g release 11.1, or later, client communicating with a server of Oracle Database 10g release 10.2, or later. For Oracle RAC environments, session pools use service metrics received from the Oracle RAC load balancing advisory¹ through Fast Application Notification (FAN) events to balance application session requests. The

work requests coming into the session pool can be distributed across the instances of Oracle RAC offering a service, using the current service performance.

Configuring OCI Clients to Receive Load Balancing Advisory FAN Events

For Oracle RAC environments, session pools use service metrics received from the Oracle RAC load balancing advisory through Fast Application Notification (FAN) events to balance application session requests. To enable your application to receive the service metrics based on the service time, ensure that you configure FAN, the load balancing advisory goal (`-rlbgoal` parameter), and the connection load balancing goal (`-clbgoal` parameter) for a service that is used by the session pool, as shown in the following example:

```
$ srvctl modify service -db crm -service ociapp.example.com -rlbgoal SERVICE_TIME  
-clbgoal SHORT -notification TRUE
```

Configuring OCI Clients to use Transaction Guard

OCI supports FAN messages and Transaction Guard. FAN is designed to quickly notify an OCI-based application of outages at the node, database, instance, service, and public network levels.

Once notified of the failure, an application can leverage Transaction Guard to reliably determine the outcome of the last in-flight transaction.

Transaction Guard avoids the costs of ambiguous errors that lead to user frustration, customer support calls, and lost opportunities. Transaction Guard is safer and performs better, with lower overheads, than home grown solutions for a known outcome.

Enabling ODP.NET Clients to Receive FAN High Availability Events

ODP.NET connection pools can subscribe to FAN HA notifications that indicate when nodes, services, and service members are down.

After a `DOWN` event, Oracle Database cleans up sessions in the connection pool that go to the instance and ODP.NET proactively removes connections that are no longer valid. ODP.NET establishes additional connections to existing Oracle RAC instances if the removal of invalid connections reduces the total number of connections to below the value for the `Min Pool Size` parameter.

When connecting to Oracle Database 12c and later, ODP.NET uses Oracle Notification Service, rather than Advanced Queuing.

Enable Fast Connection Failover for ODP.NET connection pools by subscribing to FAN high availability events. To enable Fast Connection Failover, include `"HA Events=true"` and `"pooling=true"` (the default value) in the connection string, as shown in the following example where `user_name` is the name of the database user and `password` is the password for that user:

```
con.ConnectionString =  
"User Id=user_name;Password=password;Data Source=odpnet;" +  
"Min Pool Size=10;Connection Lifetime=120;Connection Timeout=60;" +  
"HA Events=true;Incr Pool Size=5;Decr Pool Size=2";
```

¹ Run-time 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, run-time connection load balancing is a very frequent activity.

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

When connecting to Oracle Database 12c and later, ODP.NET uses Oracle Notification Service, rather than Advanced Queuing.

Use the following procedure to enable ODP.NET clients or applications to receive FAN load balancing advisory events:

1. Enable Oracle Notification Service notifications by using SRVCTL, and set the run-time load balancing goal, as shown in the following example:

```
$ srvctl modify service -db crm -service odpapp.example.com  
-notification TRUE -clbgoal LONG -rlbgoal SERVICE_TIME
```

2. Ensure Oracle Notification Service (ONS) is configured for FAN events including run-time load balancing advice.
3. To take advantage of load balancing events with ODP.NET connection pools, set the load balancing attribute in the `ConnectionString` 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.

The following example demonstrates how to configure the `ConnectionString` to enable load balancing, where `user_name` is the name of the user and `password` is the password:

```
con.ConnectionString =  
"User Id=user_name;Password=password;Data Source=odpapp;" +  
"Min Pool Size=10;Connection Lifetime=120;Connection Timeout=60;" +  
"Load Balancing=true;Incr Pool Size=5;Decr Pool Size=2";
```

Note:

ODP.NET does not support connection redistribution when a node starts (UP events). However, if you have enabled failover on the server-side, then ODP.NET can migrate connections to newly available instances.

Configuring ODP.NET Clients to use Transaction Guard

ODP.NET supports FAN messages and Transaction Guard. FAN is designed to quickly notify an ODP.NET-based application of outages at the node, database, instance, service, and public network levels.

Once notified of the failure, an application can leverage Transaction Guard to reliably determine the outcome of the last in-flight transaction.

Transaction Guard avoids the costs of ambiguous errors that lead to user frustration, customer support calls, and lost opportunities. Transaction Guard is safer and performs better, with lower overheads, than home grown solutions for a known outcome.

Distributed Transaction Processing in Oracle RAC

The X/Open Distributed Transaction Processing (DTP) architecture defines a standard architecture or interface that enables multiple application programs (APs) to share resources provided by multiple, and possibly different, resource managers (RMs). It coordinates the work between APs and RMs into global transactions.

The following sections discuss how Oracle RAC supports global (XA) transactions and DTP processing:

- [Overview of XA Transactions and Oracle RAC](#)
- [Using Global Transactions and XA Affinity for XA Transactions](#)
- [Using Services with XA Transactions on Oracle RAC](#)
- [Configuring Services for XA Applications](#)
- [Relocating Services in Administrator-Managed Databases](#)

Overview of XA Transactions and Oracle RAC

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

GTX n background processes support XA transactions in an Oracle RAC environment. The `GLOBAL_TXN_PROCESSES` initialization parameter, which is set to 1 by default, specifies the initial number of GTX n background processes for each Oracle RAC instance. Use the default value for this parameter clusterwide to allow distributed transactions to span multiple Oracle RAC instances. Using the default value 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.

Before Oracle RAC 11g release 1 (11.1), the way to achieve tight coupling in Oracle RAC was to use distributed transaction processing (DTP) services, that is, services whose cardinality (one) ensured that all tightly-coupled branches landed on the same instance—regardless of whether load balancing was enabled. If the XA application does not use suspend and resume on the same transaction branch, and does not issue savepoints that span branches, then tightly coupled XA transactions no longer require the special type of singleton services to be deployed on Oracle RAC databases. If your application cannot determine whether a transaction branch has been suspended and resumed, then the application must continue to use DTP services or preferably use XA affinity.

XA affinity (placing all branches of the same XA transaction at the same Oracle RAC instance) is a requirement when suspending and resuming the same XA branch or if using savepoints across branches. It also provides much better performance because different transactions can be balanced. XA affinity is available with Oracle WebLogic Server Active GridLink for Oracle RAC, JDBC Universal Connection Pool, and Oracle Tuxedo.

 **Note:**

Occasionally, transaction processing monitors continue to require either XA affinity or DTP services because the applications that use them suspend and resume the same branch or use save points.

An external transaction manager, such as Oracle Services for Microsoft Transaction Server (OraMTS), coordinates XA transactions. However, an internal Oracle transaction manager coordinates distributed SQL transactions. They can both use singleton services or uniform services. Singleton services provide XA affinity and can be drained for maintenance, taking advantage of global transactions.

Using Global Transactions and XA Affinity for XA Transactions

To provide improved application performance with distributed transaction processing (DTP) in Oracle RAC, you can take advantage of XA affinity.

Using XA affinity, you can direct all branches of a distributed transaction to a single instance in the cluster. To implement XA affinity you can use an application server that provides XA affinity, such as WebLogic Server and Universal Connection Pool. If your application server does not have XA affinity, then you can also use singleton services across Oracle RAC.

Connection pools at the application server tier that load balance across multiple connections to an Oracle RAC database use XA affinity to ensure that all tightly-coupled branches of a global distributed transaction run on only one Oracle RAC instance. When using a connection pool with XA affinity, your services using XA can span Oracle RAC. This is also true in distributed transaction environments using protocols such as X/Open Distributed Transaction Processing or the Microsoft Distributed Transaction Coordinator.

To enhance the performance of distributed transactions, you use services to manage DTP environments. A singleton service runs on one Oracle RAC instance at time in an Oracle RAC database. This service still allows draining for maintenance, so has better high-availability characteristics than an older DTP service. 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 it is to have one or two larger application servers. Using singleton services, global distributed transactions performed through the services 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 using global transactions
- By using more singleton services than there are Oracle RAC instances, Oracle Database can balance the load by services across all of the Oracle RAC database instances

**Note:**

The DTP service attribute or XA affinity is required, if the application suspend and resumes the same XA branch.

Using Services with XA Transactions on Oracle RAC

Most applications using XA on Oracle RAC can use uniform or (all preferred) services with XA affinity provided by the connection pool or transaction processing monitor.

The application may also use singleton services to provide XA affinity.

When using singleton services, to leverage all of the instances in a cluster, create one or more singleton services for each Oracle RAC instance that hosts distributed transactions. Choose different services for application servers 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 distributed transaction processing (DTP) transactions through multiple singleton services, thereby maximizing application throughput.

If you add or delete nodes from your cluster database, then you may have to identify and relocate services to ensure that you maintain optimum performance levels. Using singleton services, current work can complete. If you use DTP services, then current work is aborted.

You only need to use DTP services for XA applications that suspend and resume the same branch. When you are using DTP, the same approach applies as that for singletons, but you cannot drain the work when relocating services.

Configuring Services for XA Applications

To create distributed transaction processing (DTP) services for distributed transaction processing, perform the following steps:

1. Create a singleton service using Oracle Enterprise Manager or SRVCTL.

For an administrator-managed database, define only one instance as the preferred instance. You can have as many available instances as you want, for example:

```
$ srvctl add service -db crm -service xa_01.example.com -preferred RAC01  
-available RAC02,RAC03
```

For a policy-managed database, specify the server pool to use, and set the cardinality of the service to `SINGLETON`, for example:

```
$ srvctl add service -db crm -service xa_01.example.com -serverpool dtp_pool  
-cardinality SINGLETON
```

2. Set the DTP parameter (`-dtp`) for the service to `TRUE` (the default value is `FALSE`). You can use Oracle Enterprise Manager or SRVCTL to modify the DTP property of the singleton service. The following example shows how to modify the `xa_01.example.com` service using SRVCTL:

```
$ srvctl modify service -db crm -service xa_01.example.com -dtp TRUE
```

 **Note:**

If the application does require DTP services, then use the `-dtp` parameter. If not, then use the preceding example with no `-dtp` parameter.

Relocating Services in Administrator-Managed Databases

Beginning with Oracle Real Application Clusters 11g release 1 (11.1), global transactions and XA affinity replace the need for distributed transaction processing (DTP) services.

Most XA deployments should be using global transactions with XA affinity for improved load balancing and flexibility rather than the DTP attribute.

If the instance that provides a DTP service, for example `XA_01`, fails, then the service fails over in the same manner as any other service. Because there is a requirement that sessions exist on exactly one instance, however, the `-f` (force kill) option for database sessions always applies.

If services migrate to other instances, then you might have to force the relocation of the service back to the preferred instance after it is restarted to evenly re-balance the load on all of the available hardware. You can use data from the `GV$ACTIVE_SERVICES` view to determine whether you need to relocate the DTP service.

Oracle RAC Sharding

Oracle RAC Sharding affinitizes table partitions to Oracle RAC instances, and routes database requests that specify a partitioning key to the instance that logically holds the corresponding partition.

Oracle routes database requests to Oracle RAC instances in such a way that each instance always gets requests for a disjoint subset of rows in the database, which creates affinity of rows with instances. The affinity leads to higher Oracle RAC performance and scalability because of improved cache locality and reduced inter-node synchronization and block pings.

Sharding for Oracle RAC affinity uses client and server-side support for key-based routing, which is part of the Oracle Database sharding. An application that supplies a sharding key in the database using the same API implemented for sharding support in Oracle connection pools (such as Universal Connection Pool, OCI), in the same way it is done for sharding, utilizes key-based routing and, by doing so, enables Oracle RAC affinity.

Application changes that are required to supply the sharding key, do not have to affect all modules of the application. Changes can only be applied to a few frequently executed database requests. Requests that do not provide the sharding key in the connect string are routed based on the load-balancing policy. Keyless requests do not have any negative impact on data affinity because of the explicit mastership assignment of data objects to instances.

 **Note:**

Oracle only supports Oracle RAC affinity for partitioned tables. You can partition a table using any supported method without making changes to the database schema to enable this feature and then run the `ALTER SYSTEM ENABLE AFFINITY` command.

If you want to make changes to your applications to take advantage of affinity-enabling routing, then you may also take advantage of sharding when data is distributed across multiple independent databases. You can later move to distributed sharding if you require extreme scalability or fault isolation.

Automatic Workload Repository

The Automatic Workload Repository (AWR) collects, processes, and maintains performance statistics for the database.

The gathered data can be displayed in both reports and views. If you use services with your database, then AWR tracks [metrics](#) at the service level.

Metrics can be measured against a variety of units, including time, transactions, or database calls. For example, the number of database calls per second is a metric. Server generated alerts can be placed on these metrics when they exceed or fail to meet user-specified thresholds. The database or system administrator can then respond, for example, by:

- Using the Oracle Database Resource Manager to configure the service level for one service to have priorities relative to other services
- Stopping overloaded processes
- Modifying a service level requirement
- Implementing recovery scenarios in response to service quality changes

Using AWR metrics and performance alerts enables you to maintain continued service availability despite service level changes. It also enables you to measure the quality of service provided by the database services.

The AWR ensures that the Oracle Clusterware workload management framework and the database resource manager have persistent and global representations of performance data. This information helps Oracle Database schedule job classes by service and to assign priorities to consumer groups. If necessary, you can rebalance workloads manually with either Oracle Enterprise Manager or SRVCTL. You can also disconnect a series of sessions, but leave the service running.

 **Note:**

Oracle *does not* recommend using the `DBMS_SERVICE` package for use with services used by an Oracle RAC database. Use SRVCTL or Oracle Enterprise Manager to create database services for Oracle RAC.

Measuring Performance by Service Using the Automatic Workload Repository

Services add a new dimension for performance tuning because workloads are visible and measurable, and therefore 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 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 Database 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_MONITOR` PL/SQL package. For example, for connections that use the `erp` service, the following command enables monitoring for the `exceptions pay` action in the `payroll` module:

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

For connections that use the `erp` service, the following command enables monitoring for all actions in the `payroll` module:

```
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 for application modules and actions.

Statistics aggregation and tracing by service are global in scope for Oracle RAC databases. In addition, these statistic aggregations are persistent across instance restarts and service relocations for both Oracle RAC and noncluster 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$SERVICE_EVENT`, `V$SERVICE_WAIT_CLASS`, `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 for a five second interval. 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
    , callsperssec 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 ;

```

Automatic Workload Repository Service Thresholds and Alerts

Service level thresholds enable you to compare actual service levels against required levels of service. This provides accountability for 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.

AWR enables you to explicitly specify two performance thresholds for each service: the response time for calls (`ELAPSED_TIME_PER_CALL`), and the CPU time for calls (`CPU_TIME_PER_CALL`). The response time threshold indicates that the elapsed time for each user call for each service should not exceed a certain value, and the CPU time for calls threshold indicates that the time spent using the CPU for each call for each service should not exceed a certain value. 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.

You must set these thresholds on each instance of an Oracle RAC database. The elapsed time and CPU time are calculated as the moving average of the elapsed, server-side call time. The AWR monitors the elapsed time and CPU time and publishes AWR alerts when the performance exceeds the thresholds. You can schedule actions using Oracle Enterprise Manager jobs for these alerts, or you can schedule actions to occur programmatically when the alert is received. You can respond to these alerts by changing the priority of a job, stopping overloaded processes, or by relocating, starting or stopping a service. This permits you to maintain service availability despite changes in demand.

This section includes the following topics:

- [Example of Services and Thresholds Alerts](#)
- [Enable Service_ Module_ and Action Monitoring](#)

Example of Services and Thresholds Alerts

In this scenario, you need to check the thresholds for the payroll service. You can use the AWR report to get this information. You should compare the results from reports run over several successive intervals during which time the system is running optimally. For example, assume that for servers accessed by a payroll application, the AWR report runs each Thursday during the peak usage times of 1:00 p.m. to 5:00 p.m. The AWR report contains the response time, or elapsed database time, and the CPU consumption time, or CPU time, for calls for each server, including the `payroll` service. The AWR report also provides a breakdown of the work done and the wait times that are contributing to the response times.

Using `DBMS_MONITOR`, you set a warning threshold for the elapsed time per call for the `payroll` service at 0.5 seconds (500000 microseconds). You also set a critical threshold for the elapsed time per call for the `payroll` service at 0.75 seconds (750000 microseconds).

In this example, thresholds are added for the `payroll` service 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 => 'payroll');
```

You can verify the threshold configuration is set on all the instances 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. For example, you might decide to set the following:

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

The following commands show how to enable the module and action monitoring for the services:

```
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');  
EXECUTE DBMS_MONITOR.SERV_MOD_ACT_STAT_ENABLE(service_name => 'hot_batch',  
module_name => 'posting');
```


To verify monitoring is enabled for the service, module, and actions, use 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 is similar to the following:

AGGREGATION	SERVICE	MODULE	ACTION
-----	-----	-----	-----
SERVICE_MODULE_ACTION	erp	payroll	exceptions pay
SERVICE_MODULE	erp	payroll	
SERVICE_MODULE	hot_batch	posting	

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 use one service, while batch processing can use another and reporting can use yet another service to connect to the database.

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 represent 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.

Service Deployment Options

This section describes the following service deployment topics:

- [Service Usage in an Oracle RAC Database](#)
- [Service Characteristics](#)
- [Default Service Connections](#)
- [Restricted Service Registration](#)

Service Usage in an Oracle RAC Database

Services provide location transparency. A service name can identify multiple database instances, and an instance can belong to multiple services. Several database features use services for an Oracle RAC database:

This section includes the following topics:

- [Oracle Clusterware Resources for a Service](#)
- [Database Resource Manager Consumer Group Mappings for Services](#)
- [Performance Monitoring by Service with AWR](#)
- [Parallel Operations and Services](#)
- [Oracle Streams and Oracle RAC](#)

Oracle Clusterware Resources for a Service

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.

Database Resource Manager Consumer Group Mappings for Services

Services are integrated with Oracle Resource Manager, which enables you to restrict the resources that users use to connect to an instance by using a service. Oracle Resource Manager enables you to map a consumer group to a service so that users who connect to an instance using that service are members of the specified consumer group. Oracle Resource Manager operates at an instance level.

Performance Monitoring by Service with AWR

The metric data generated by Automatic Workload Repository (AWR) is organized into various groups, such as event, event class, session, service, and tablespace metrics. Typically, you view the AWR data using Oracle Enterprise Manager or AWR reports.

Parallel Operations and Services

By default, in an Oracle RAC environment, a SQL statement executed in parallel can run across all of the nodes in the cluster. For this cross-node or inter-node parallel execution to perform well, the interconnect in the Oracle RAC environment must be sized appropriately because inter-node parallel execution may result in a lot of interconnect traffic. To limit inter-node parallel execution, you can control parallel execution in an Oracle RAC environment using the `PARALLEL_FORCE_LOCAL` initialization parameter. By setting this parameter to `TRUE`, the parallel execution servers can only execute on the same Oracle RAC node where the SQL statement was started.

Services are used to limit the number of instances that participate in a parallel SQL operation. When the default database service is used, the parallel SQL operation can run on all available instances. You can create any number of services, each consisting of one or more instances. When a parallel SQL operation is started, the parallel execution servers are only spawned on instances which offer the specified service used in the initial database connection.

`PARALLEL_INSTANCE_GROUP` is an Oracle RAC parameter that, when used with services, lets you restrict parallel query operations to a limited number of instances. To restrict parallel query operations to a limited number of instances, set the `PARALLEL_INSTANCE_GROUP` initialization parameter to the name of a service. This does not affect other parallel operations such as parallel recovery or the processing of `GV$` queries.

Oracle Streams and Oracle RAC

Oracle Streams takes advantage of Oracle RAC features.

When Oracle Streams is configured in an Oracle RAC environment, each queue table has an owning instance. If the instance that hosts a queue table fails, another instance

in the Oracle RAC database becomes the owning instance for the queue table, allowing Oracle Streams to continue operating.

Also, on an Oracle RAC database, a service is created for each buffered queue. This service always runs on the owner instance of the destination queue and follows the ownership of this queue if the ownership switches because of instance startup, instance shutdown, and so on. This service is used by queue-to-queue propagations.

Service Characteristics

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

- [Service Name](#)
- [Service Edition](#)
- [Service Management Policy](#)
- [Database Role for a Service](#)
- [Instance Preference](#)
- [Server Pool Assignment](#)
- [Load Balancing Advisory Goal for Run-time Connection Load Balancing](#)
- [Connection Load Balancing Goal](#)
- [Distributed Transaction Processing](#)
- [Failing Over OCI Clients with TAF](#)

Service Name

Each service has a service name. The service name is used by clients to connect to one or more instances. The service name must be unique throughout your system.

Service Edition

Edition-based redefinition of database objects enables you to upgrade an application's objects while these objects are in use. You can set the *edition* attribute of a database service when you create it, or modify an existing service to set the edition. When you set the service edition, connections that use this service use this edition as the initial session edition. If the service does not specify the edition name, then the initial session edition is the database default edition.

You can set the service edition using SRVCTL, as follows:

```
$ srvctl modify service -db hr -s crmsrv -edition e2
```


Service Management Policy

When you use Oracle Clusterware to manage your database, you can configure startup options for each individual database service when you add the service using the `srvctl add service` command with the `-policy` parameter.

If you set the management policy for a service to `AUTOMATIC` (the default), then the service starts automatically when you start the database with SRVCTL. If you set the

management policy to `MANUAL`, then the service does not automatically start, and you must manually start it with `SRVCTL`. A `MANUAL` setting does not prevent Oracle Clusterware from monitoring the service when it is running and restarting it if a failure occurs. Before Oracle RAC 11g release 2 (11.2), all services worked as though they were defined with a `MANUAL` management policy.

Using `CRSCTL` to stop and restart Oracle Clusterware is treated as a failure and the service is restarted if it was previously running.

 **Note:**

When you use automatic services in an administrator-managed database, during planned database startup, services may start on the first instances to start rather than their preferred instances, provided that the started instances are in the (combined) preferred and available services list.

Database Role for a Service

If you configured Oracle Data Guard in your environment, then you can define a role for services when you add or modify a service using `SRVCTL` and the `-role` parameter with the appropriate command.

When you specify a role for a service, Oracle Clusterware automatically starts the service only when the database role matches the role you specified for the service. Valid roles are `PRIMARY`, `PHYSICAL_STANDBY`, `LOGICAL_STANDBY`, and `SNAPSHOT_STANDBY` and you can specify more than one role for a service.

 **Note:**

The service role only controls automatic startup of services. Using `SRVCTL` to manually start a service will succeed even if the roles do not match.

Redo Apply (physical standby database) can run on all or some standby instances that you can configure. This enables Redo Apply performance to scale, if necessary, by adding additional standby instances.

If multiple databases in the cluster offer the same service name, then Oracle RAC balances connections to that service across all such databases. This is useful for standby and active Oracle Data Guard databases, but if you want client connections to a service to be directed to a particular database, then the service name must be unique within the cluster (not offered by any other database).

Instance Preference

When you define a service for an administrator-managed database, you define which instances normally support that service using `SRVCTL` with the `-preferred` parameter.

These are known as the *preferred* instances. You can also define other instances to support a service if the service's preferred instance fails using `SRVCTL` with the `-available` parameter. These are known as *available* instances.

When you specify preferred instances, you are specifying the number of instances on which a service normally runs. This is the *maximum cardinality* of the service. 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.

If an instance fails, then you cannot control to which available instance Oracle Clusterware relocates the services if there are multiple instances in the list. During a planned operation, however, you can manually direct the service to any instance in either the preferred or the available list not currently offering the service.

When a service moves to an available instance, Oracle Database does not move the service back to the preferred instance when the preferred instance restarts because:

- The service is 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 to the preferred instance by using FAN callouts.

Server Pool Assignment

When you define services for a policy-managed database, you assign the service to a server pool in which the database is hosted using SRVCTL with the `-serverpool` parameter.

You can define the service as either `UNIFORM` (running on all instances in the server pool) or `SINGLETON` (running on only one instance in the server pool) using the `-cardinality` parameter. For singleton services, Oracle RAC chooses on which instance in the server pool the service is active. If that instance fails, then the service fails over to another instance in the server pool. A service can only run in one server pool and Oracle recommends that every server pool has at least one service.

Note:

Oracle Database Quality of Service Management (Oracle Database QoS Management) manages singleton services in a server pool, if the maximum size of that server pool is one.

Load Balancing Advisory Goal for Run-time Connection Load Balancing

With run-time connection load balancing, applications can use load balancing advisory events to provide better service to users. Oracle JDBC, Oracle Universal Connection Pool for Java, OCI session pool, ODP.NET, and Oracle WebLogic Server Active GridLink for Oracle RAC 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. To enable the load balancing advisory, use SRVCTL with the `-r1bgoal` parameter when creating or modifying the service.

The load balancing advisory also recommends how much of the workload should be sent to that instance. The goal 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).

Connection Load Balancing Goal

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 use SRVCTL to define the method you want the listener to use for load balancing by setting the connection load balancing goal, specified with the `-clbgoal` parameter. Connections are classified as `LONG` (such as connection pools and `SQL*FORMS`), which tells the listener to use session count, or `SHORT`, which tells the listener to use response-time or throughput statistics.

If load balancing advisory is enabled (the `-rlbgoal` parameter *does not* equal `NONE`), then connection load balancing attempts to use load balancing advisory (whether it is set to `SHORT` or `LONG`). If load balancing advisory is set to `SHORT`, then it will use the `GOODNESS` value of a service to try to prevent all connection requests from going to one instance. If load balancing advisory is set to `LONG`, then it uses `run queue length` if the service is non-uniform, or `session count` if there is uniform service distribution.

Distributed Transaction Processing

[Oracle XA](#) applications have unique requirements. Oracle provides global transactions across Oracle RAC. For best performance, use XA affinity (all branches at the same instance) for most transactions, and global transactions when needed. You can use XA affinity with connection pools, such as Universal Connection Pools and WebLogic Server. You can also use singleton services that you create using SRVCTL. Use SRVCTL, as well, to set the distributed transaction processing parameter (`-dtp`) to `TRUE` if you are suspending and resuming the same Oracle XA branch. Do not use this in general, however, because it does not offer rolling planned maintenance.

Default Service Connections

Your Oracle RAC database includes an Oracle database service identified by `DB_UNIQUE_NAME`, if set, or `DB_NAME` or `PDB_NAME`, if not. 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. Additionally, the database 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

All of these services are used for internal management. You cannot stop or disable any of these internal services to do planned outages or to failover to Oracle Data Guard. Do not use these services for client connections.

 **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.

Restricted Service Registration

This feature allows listener registration only from local IP addresses, by default, and provides the ability to configure and dynamically update a set of IP addresses or subnets from which registration requests are allowed by the listener.

Security is a high priority to all enterprises, and network security and controlling access to the database is a critical component of overall security endeavours. Database Instance registration with a listener succeeds only when the request originates from a valid node. The network administrator can specify a list of valid nodes, excluded nodes, or disable valid node checking. The list of valid nodes explicitly lists the nodes and subnets that can register with the database. The list of excluded nodes explicitly lists the nodes that cannot register with the database. The control of dynamic registration results in increased manageability and security of Oracle RAC deployments.

By default, valid node checking for registration (VNCR) is enabled. In the default configuration, registration requests are only allowed from nodes within the cluster, because they are redirected to the private subnet, and only nodes within the cluster can access the private subnet. Non-SCAN listeners only accept registration from instances on the local node. You must manually include remote nodes or nodes outside the subnet of the SCAN listener on the list of valid nodes by using the `registration_invited_nodes_alias` parameter in the `listener.ora` file or by modifying the SCAN listener using SRVCTL, as follows:

```
$ srvctl modify scan_listener -invitednodes node_list -invitedsubnets subnet_list
```

Administering Services

You can create and administer services with Oracle Enterprise Manager and the SRVCTL utility. The following sections describe how to perform service-related tasks using these tools:

This section includes the following topics:

- [Overview of Service Administration](#)
- [Administering Services with Oracle Enterprise Manager](#)
- [Administering Services with SRVCTL](#)

 **Note:**

You can also use the `DBMS_SERVICE` package to create or modify services and service attributes, but `SRVCTL` will override any settings made using this package. The `DBMS_SERVICE` package is not recommended for use with services used by an Oracle RAC database, nor when Oracle Restart is used, nor when Oracle Clusterware is managing a single-instance database.

Overview of Service Administration

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. Oracle Database can measure the performance for each service. Using the `DBMS_MONITOR` package, you can define both the application modules within a service and the individual actions for a module and monitor thresholds for these actions, enabling you to manage workloads to deliver capacity on demand.

When you create new services for your database, you should define the automatic workload management characteristics for each service, as described in "[Service Characteristics](#)".

 **See Also:**

Oracle Database Quality of Service Management User's Guide if you are using Oracle Database QoS Management with your Oracle cluster for details on how to configure the database services

In addition to creating services, you can:

- **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 have to check the status on an instance or service basis. For example, you may have 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 shut down your database, for example, by running the `SRVCTL` command `srvctl stop database -db db_unique_name` where `db_unique_name` is the name of the database you want to stop, then Oracle Database stops all services for that database. Depending on the service management policy, you may have to manually restart the services when you start the database. Both the `srvctl stop database` and `srvctl stop service`

commands accept the `-force` option to forcibly disconnect connections. To drain sessions for planned outages do not use the `-force` option.

 **Note:**

If Oracle Database QoS Management is enabled for the Oracle RAC database, then the services are automatically restarted after they are stopped.

- **Map a service to a consumer group.** You can 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.
- **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 must perform database or instance maintenance, such as when you are performing an upgrade and you want to prevent connection requests from succeeding.
- **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.

 **Note:**

- When you use services, do not set a value for the `SERVICE_NAMES` parameter; Oracle Database 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 Database provides when you set `SERVICE_NAMES`. In addition, setting a value for this parameter may override some benefits of using services.
- Service status information must be obtained from `SRVCTL` or from the service-related database views, such as `dba_services`.
- 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 does not stop users connecting with the service.)

Administering Services with Oracle Enterprise Manager

The Cluster Managed Database Services page is the master page for beginning all tasks related to services.

Access this page, as follows:

1. In Oracle Enterprise Manager, go to the Cluster Database Home page.

 **See Also:**

Oracle Database 2 Day DBA for details on logging in to Oracle Enterprise Manager.

2. From the **Availability** menu, select **Cluster Managed Database Services** to display the Cluster Managed Database Services page.
3. Enter or confirm the credentials for the Oracle RAC database and host operating system and click **Continue** to display the Cluster Managed Database Services page.

From the Cluster Managed Database Services page you can drill down to perform the following tasks:

- View a list of services for the cluster
- View the instances on which each service is currently running
- View the server pool and nodes offering the service in a policy-managed environment
- 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

 **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`.

Administering Services with SRVCTL

When you create a service using SRVCTL, you must start the service with a separate SRVCTL command.

However, you may later have to manually stop or restart the service. You may also have 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](#)
- [Creating Services for Application Continuity and Transaction Guard](#)
- [Starting and Stopping Services with SRVCTL](#)
- [Enabling and Disabling Services with SRVCTL](#)

- [Relocating Services with SRVCTL](#)
- [Obtaining the Status of Services with SRVCTL](#)
- [Obtaining the Configuration of Services with SRVCTL](#)

Creating Services with SRVCTL

To create a service with SRVCTL, use the `srvctl add service` command on the command line.

Creating Services for Application Continuity and Transaction Guard

To configure services for Application Continuity, when you create a service using SRVCTL, set the `-failovertype` parameter to `TRANSACTION` and `-commit_outcome` to `TRUE`.

When using Application Continuity and Transaction Guard with your applications, you must configure a service. This section describes how to configure these application services depending on the functionality you plan to implement.

Creating Services for Application Continuity

Additionally, you can set values for these other service parameters for Application Continuity and load balancing:

- `-replay_init_time`: Specifies how long, in seconds, you allow replay to start. Oracle recommends that you choose a value based on how long you will allow replay to be initiated. The default value is 300 seconds.
- `-retention`: Specifies the time (in seconds) that the commit outcome information is stored in the database. The default value is 86400 (1 day).
- `-session_state`: After a COMMIT has executed, if the state was changed in that transaction, then it is not possible to replay the transaction to reestablish that state if the session is lost. When configuring Application Continuity, the applications are categorized depending on whether the session state after the initial setup is dynamic or static, and then whether it is correct to continue past a COMMIT operation within a request.
 - **Dynamic**: (default) A session has a dynamic state if the session state changes are not fully encapsulated by the initialization, and cannot be fully captured in a callback at failover. Once the first transaction in a request commits, failover is internally disabled until the next request begins. This is the default mode that almost *all* applications should use for requests.
 - **Static**: (special—on request) A session has a static state if all session state changes, such as NLS settings and PL/SQL package state, can be repeated in an initialization callback. This setting is used only for database diagnostic applications that do not change session state. Do not specify `STATIC` if there are any non-transactional state changes in the request that cannot be reestablished by a callback. If you are unsure what state to specify, use `DYNAMIC`.
- `-failoverretry`: Number of connection retries for each connection attempt; recommended value is 30.
- `-failoverdelay`: Delay in seconds between each connection attempt; recommended value is 10.

- `-notification`: FAN is highly recommended—set this value to `TRUE` to enable FAN for OCI and ODP.Net clients.
- `-clbgoal`: For connection load balancing, use `SHORT` when using run-time load balancing.
- `-rlbgoal`: For run-time load balancing, set to `SERVICE_TIME`.

To create a service for Application Continuity for a policy-managed Oracle RAC database, use a command similar to the following, where `racdb` is the name of your Oracle RAC database, `app2` is the name of the service you are modifying, and `Srvpool1` is the name of the server pool in which the service is offered:

```
$ srvctl add service -db racdb -service app2 -serverpool Srvpool1
  -failovertype TRANSACTION -commit_outcome TRUE -replay_init_time 1800
  -retention 86400 -notification TRUE -rlbgoal SERVICE_TIME -clbgoal SHORT
  -failoverretry 30 -failoverdelay 10
```

You can use `SRVCTL` to modify an existing service for Application Continuity, similar to the following command, where `racdb` is the name of your Oracle RAC database, and `app1` is the name of the service you are modifying:

```
$ srvctl modify service -db racdb -service app1 -clbgoal SHORT
  -rlbgoal SERVICE_TIME -failoverretry 30 -failoverdelay 10
  -failovertype TRANSACTION -commit_outcome TRUE -replay_init_time 1800
  -retention 86400 -notification TRUE
```

Creating Services for Transaction Guard

To enable Transaction Guard, but not Application Continuity, create the service using `SRVCTL` and set only `-commit_outcome TRUE`.

You can use `SRVCTL` to modify an existing service to enable Transaction Guard, similar to the following command, where `racdb` is the name of your Oracle RAC database, and `app2` is the name of the service you are modifying:

```
$ srvctl modify service -db racdb -service app2 -commit_outcome TRUE
  -retention 86400 -notification TRUE
```

In the preceding example, the `-retention` parameter specifies how long, in seconds, to maintain the history. Additionally the `-notification` parameter is set to `TRUE`, enabling FAN events.

To use Transaction Guard, a DBA must grant permission, as follows:

```
GRANT EXECUTE ON DBMS_APP_CONT;
```

Starting and Stopping Services with SRVCTL

For applications to connect using a server, the service must be started. If you stop a service, then it is temporarily unavailable, but is still subject to automatic restart and failover.

Enter the following `SRVCTL` syntax at the command line to start or stop a service:

```
$ srvctl start service -db db_unique_name [-service service_name_list]
  [-instance inst_name] [-startoption start_options]

$ srvctl stop service -db db_unique_name -service service_name_list
  [-instance inst_name] [-startoption start_options]
```

Enabling and Disabling Services with SRVCTL

If you disable a service, then Oracle Clusterware does not consider the service for automatic startup, failover, or restart. You might disable a service when performing application maintenance, to ensure the service is not accidentally restarted by Oracle Clusterware until your maintenance operations are complete. To make the service available for normal operation again, you enable the service.

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

```
$ srvctl enable service -db db_unique_name -service service_name_list
  [-instance inst_name]
```

```
$ srvctl disable service -db db_unique_name -service service_name_list
  [-instance inst_name]
```

Relocating Services with SRVCTL

Run the `srvctl relocate service` command from the command line to relocate a service. You might use this command when a service has failed over to an available instance, but you want to move it back to the preferred instance after that instance is restarted.

The following command relocates the `crm` service from instance `apps1` to instance `apps3`:

```
$ srvctl relocate service -db apps -service crm -oldinst apps1 -newinst apps3
```

The following command relocates the `crm` service from `node1` to `node3` using node syntax:

```
$ srvctl relocate service -db apps -service crm -currentnode node1
  -targetnode node3
```

Obtaining the Status of Services with SRVCTL

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

```
$ srvctl status service -db apps

Service erp is running on nodes: apps02,apps03
Service hr is running on nodes: apps02,apps03
Service sales is running on nodes: apps01,apps04
```

Obtaining the Configuration of Services with SRVCTL

Run the `srvctl config 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 `erp` service that is running on the `apps` database:

```
$ srvctl config service -db apps -service erp

Service name: erp
Service is enabled
```

```
Server pool: pool1
Cardinality: 1
Disconnect: false
Service role: PRIMARY
Management policy: AUTOMATIC
DTP transaction: false
AQ HA notifications: true
Global: false
Commit Outcome: true
Failover type: TRANSACTION
Failover method: NONE
TAF failover retries: 30
TAF failover delay: 10
Connection Load Balancing Goal: LONG
Runtime Load Balancing Goal: SERVICE_TIME
TAF policy specification: NONE
Edition:
Pluggable database name:
Maximum lag time: ANY
SQL Translation Profile:
Retention: 86400 seconds
Replay Initiation Time: 1800 seconds
Session State Consistency: STATIC
Preferred instances: apps
Available instances:
```

Global Services

Oracle RAC supports database services and enables service-level workload management across instances in a single cluster.

Global services provide dynamic load balancing, failover, and centralized service management for a set of replicated databases that offer common services. The set of databases may include Oracle RAC and non-clustered Oracle databases interconnected by Oracle Data Guard, Oracle GoldenGate, or any other replication technology.

When you create and use global services, the following workload management features are available:

- Ability to specify preferred and available databases for a global service
- Handling of replication lag
- Geographical affinity between clients and servers
- Connection load balancing
- Run-time load balancing
- Inter-database service failover
- Fast connection failover
- Connect-time failover
- Application Continuity
- Transaction Guard
- Backward compatibility with existing clients

**Note:**

You can manage instance placement of a global service within an Oracle RAC database with SRVCTL but you can only manage other global service attributes with GDSCTL.

Service-Oriented Buffer Cache Access

Service-oriented buffer cache access improves performance by managing data with the service to which the data belongs.

Access of an object, over time, through a service is mapped and persisted to the database, and this information can be used to improve performance. Blocks that are accessed through the service are cached in the instances where the services are running and, more importantly, the information is not cached where the services are not running.

This information can also be used to pre-warm the cache prior to a service starting. The service start-up can be triggered either by instance start-up or by service relocation. Service-oriented buffer cache access provides consistent performance to any user of that service because the blocks that the service user accesses are cached in the new relocated instance.

Connecting to a Service: An Example

The following example illustrates how to create a service and then gives several examples of connecting to that service using different client methods.

In this example the service is enabled for run-time load balancing, as follows:

- Service Name: `HR.example.com`
 - Running on database named CRM
 - The system consists of 4 nodes
- Specifying `SERVICE_TIME` as the value for the `-rlbgoal` parameter
- SCAN address of the listener is `rws3010104-scan.example.com`
- Listener port is 1585

The service has a cardinality of two, but if needed, can be offered by any of the CRM database instances. The service configuration is as follows:

- Preferred Instances: CRM1, CRM2
- Available Instances: CRM3, CRM4
- Specifying `SHORT` as the value for the `-clbgoal` parameter

The application using this service takes advantage of Application Continuity, so you must set `-failovertype` and `-commit_outcome`. Use the default retention parameters, but set a 10 second delay between connection attempts, and up to 40 retries before failing to get a connection.

Creating the HR Service Using SRVCTL

Create the HR service using SRVCTL, as follows:

```
$ srvctl add service -db CRM -service HR.example.com -preferred CRM1,CRM2
  -available CRM3,CRM4 -clbgoal SHORT -failovertime TRANSACTION
  -commit_outcome TRUE -failoverdelay 10 -failoverretry 40
```

Start the HR.example.com service, as follows:

```
$ srvctl start service -db CRM -service HR.example.com
```

The service is now be available on up to two instances, and CRM1 and CRM2 are the preferred instances.

Connecting to the HR Service from a JDBC Application

The application that connects to the HR service, in this example, is a JDBC application using the JDBC Universal Connection Pool with the JDBC thin driver.

In this example, a URL is constructed specifying the thin-style service name format for the database specifier. Fast Connection Failover is enabled, and remote Oracle Notification Service is configured, where the Oracle Notification Service daemon on the cluster listens on port 6200.

```
//import packages and register the driver
import java.sql.Connection;
import java.sql.SQLException;
import java.sql.Statement;
import oracle.ucp.jdbc.PoolDataSourceFactory;
import oracle.ucp.jdbc.PoolDataSource;

PoolDataSource pds = PoolDataSourceFactory.getPoolDataSource();

//set the connection properties on the data source.
pds.setConnectionPoolName("FCFPool");
pds.setFastConnectionFailoverEnabled(true);
pds.setONSConfiguration("nodes=rws3010104-scan.example.com:6200");
pds.setConnectionFactoryClassName("oracle.jdbc.pool.OracleDataSource");
pds.setURL("jdbc:oracle:thin:@//rws3010104-scan.example.com:1585/HR.example.com");
pds.setUser("HR");
pds.setPassword("hr");

//Override any pool properties.
pds.setInitialPoolSize(5);

//Get a database connection from the datasource.

Connection conn = pds.getConnection();

// do some work

//return connection to pool
conn.close();
conn=null
```


6

Ensuring Application Continuity

Application Continuity is a feature that enables the replay, in a non-disruptive and rapid manner, of a request against the database after a recoverable error that makes the database session unavailable so an outage appears to the user as no more than a delayed execution of the request.

The request can contain transactional and non-transactional work. After a successful replay, the application can continue where that database session left off, instead of having users left in doubt, not knowing what happened to their funds transfers, flight bookings, and so on, and avoiding the need to reboot mid-tier servers to recover from an overload of log ins when the application comes back online. With Application Continuity, the end-user experience is improved by masking many outages, planned and unplanned, without the application developer needing to attempt to recover the request.

Without Application Continuity, it can be almost impossible for an application to mask outages in a safe way, for reasons that include the following:

- The state at the client remains at present time, with entered data, returned data, and variables cached.
- If a `COMMIT` has been issued, then the `COMMIT` failure message cannot be retrieved if it is not received by the client or the application.
- Checking the status of an in-doubt transaction at a point in time is no guarantee that it will not `COMMIT` later.
- Non-transactional database session state that the application needs to operate is lost.
- If the request can continue, the database and the database session must be in the right state.

With Application Continuity, however, Oracle Database, the Oracle drivers, and the Oracle connection pools all collaborate to mask many outages in a safe and reliable way.

Application Continuity improves developer productivity by attempting to mask outages that can be masked. However, applications still must include error handling for these cases:

- *Nonrecoverable* errors, such as invalid input data. (Application Continuity applies only to recoverable errors.)
- Recoverable errors when replay has encountered a restriction, such as use of concrete classes in the application, or when replay has not been able to restore the client-visible state to that on which the client may have made decisions so far.

Introduced in Oracle Database 12c Release 1 (12.1.0.1), Application Continuity strengthens the fault tolerance of systems and applications that use an Oracle database.

This chapter assumes that you are familiar with the major relevant concepts and techniques of the technology or product environment in which you are using Application Continuity, such as Oracle WebLogic Server, Oracle RAC, or Oracle Active Data Guard (Oracle ADG).

This chapter includes the following topics:

- [Fast Application Notification](#)
- [Managing Unplanned Outages](#)
- [Managing Planned Maintenance](#)
- [About Application Continuity](#)
- [Application Continuity Operation and Usage](#)
- [Potential Side Effects of Application Continuity](#)
- [Restrictions and Other Considerations for Application Continuity](#)
- [Transaction Guard for Improving Client Failover](#)
- [Failing Over OCI Clients with TAF](#)

Fast Application Notification

The Oracle RAC high availability framework monitors a database and its services and sends event notifications using Fast Application Notification (FAN).

Oracle Database focuses on maintaining the highest possible service availability. In Oracle RAC, 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. Oracle Clusterware recovers and balances services according to business rules and the service attributes.

This section includes the following topics:

- [Overview of Fast Application Notification](#)
- [Fast Application Notification High Availability Events](#)
- [Subscription to High Availability Events](#)
- [Using Fast Application Notification Callouts](#)

Overview of Fast Application Notification

FAN provides immediate interrupt of clients following outages related to the database, nodes, and networks.

FAN is essential to break clients out of TCP/IP timeouts immediately following failures. FAN notifies clients immediately when resources become available and initiates draining of database sessions so clients experience no outages during planned maintenance. FAN also includes notifying configuration- and service-level information that includes changes in service status.

The Oracle client drivers and Oracle connection pools respond to FAN events and take immediate action. FAN UP and DOWN events apply to services, databases, instances, networks, and nodes.

 **Note:**

FAN is supported from Oracle Database 10g release 2 (10.2), on.

Oracle connection pools, for example, use FAN to receive very fast notification of failures, to balance connections following failures, and to balance connections again after the failed components are repaired. So, when a service at an instance starts, the connection pool uses the FAN event to route work to that resource, immediately. When a service at an instance or node fails, the connection pool uses the FAN event to immediately interrupt applications to recover. FAN is essential to prevent applications from hanging on TCP/IP timeouts.

Importance of FAN

Applications can waste time in many critical ways:

- Waiting for TCP/IP timeouts when a node fails without closing sockets, and for every subsequent connection while that IP address is down.
- Attempting to connect when services are down.
- Not connecting when services resume.
- Processing the last result at the client when the server goes down.
- Attempting to execute work on sub-optimal nodes.

When a node fails without closing sockets, all sessions that are blocked in an I/O wait (read or write) wait for `tcp_keepalive`. This wait status is the typical condition for an application connected by a socket. Sessions processing the last result are even worse off, not receiving an interrupt until the next data is requested. Using FAN events eliminates applications waiting on TCP timeouts, time wasted processing the last result at the client after a failure has occurred, and time wasted executing work on slow, hung, or dead 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 time out against the database and detect a problem, applications can receive FAN events and react immediately. With FAN, in-flight transactions are immediately terminated and the client notified when the instance fails.

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.

Oracle Database 12c release 2 (12.2) client drivers are FAN-aware, and FAN is enabled, by default. This includes the JDBC Thin driver (12.2.0.1) and Oracle Data Provider for Net (ODP.NET) drivers. A client driver can detect planned and unplanned FAN events and take action beneath the application.

For planned maintenance and applications using OCI or Pro* (and not using the OCI session pool or Tuxedo), an application must check `OCI_ATTR_SERVER_STATUS`. Add this check when sessions are returned to your own connection pool, and for idle connections, regularly.

Following a FAN down event with planned maintenance, this attribute is set to `OCI_SERVER_NOT_CONNECTED`. The application closes the connection after reading this

disconnected status. The session remains open for draining of active work until the application closes, providing error-free failover.

You can take advantage of FAN events in the following ways:

- Applications can use FAN without programmatic changes if you use an integrated Oracle client. The integrated clients for FAN events include Oracle JDBC Universal Connection Pool, ODP.NET connection pool, OCI session pool, Oracle WebLogic Server Active Gridlink for Oracle RAC, and OCI and ODP.NET clients. The integrated Oracle clients must be Oracle Database 10g release 2 (10.2) or later to take advantage of the FAN high-availability events. The pooled clients can also take advantage of the load balancing advisory FAN events.
- You can configure third-party application containers, such as those provided by Apache Tomcat and WebSphere, to use the built-in FAN support offered by using the Universal Connection Pool in place of the default pool, which is certified as a connection pool for third-party Java application servers including Apache Tomcat and WebSphere.
- Use the FAN-aware capability of the Oracle drivers by using standard interfaces to test connections on *get* or *release* from the third-party connection pools in use by third-party application servers or custom applications.
 - This solution applies to standard Java applications through the use of the standard TNS connect string and ensures that the `ons.jar` and `simpleFAN.jar` files are available on the application CLASSPATH.
 - For the OCI/OCID driver, the `OCI_ATTR_SERVER_STATUS` server context handle attribute is sensitive to FAN events and will return `OCI_SERVER_NOT_CONNECTED` if the connection has been affected by a FAN event.
- You can implement FAN with server-side callouts on your database tier.
- Applications can use FAN programmatically by using the JDBC and Oracle RAC FAN application programming interface (API) or by using callbacks with OCI and ODP.NET to subscribe to FAN events and to run event handling actions upon the receipt of an event.

If you use one of the integrated clients listed in the first item of the preceding list, then, for DOWN events, the disruption to the application is minimized because the FAN-aware client terminates the connections to the failed instance or node before they are reused. Active work can be allowed to complete and, if there is a surviving instance, then continuous service can be maintained for ongoing work. Any sessions active when the instance or service stops are terminated and the application user is immediately notified. Incomplete transactions can be protected by Application Continuity, if it is enabled. Application users who request connections are directed to available instances, only.

For UP events, when services and instances are started, new connections are created so that the application can immediately take advantage of the extra hardware resources or additional capacity.

To take advantage of FAN-aware capabilities in the drivers, as mentioned in the previous list, the following is required:

- For the thin Java driver, beginning with release 12.2, FAN is automatically enabled by placing the `ons.jar` and `simpleFAN.jar` files on the CLASSPATH, and by using the recommended TNS format (as shown in [Example 6-1](#)). Using the recommended TNS format automatically configures ONS. Also with the Java thin driver, FAN is supported for both planned and unplanned events. For unplanned

outages, the FAN interrupt is immediate. For planned maintenance, configure the Java application servers or custom pools using standard interfaces to test connections on *get* or *release* from third-party connection pools. For example, `TestConnectionsOnReserve`, `TestOnBorrow`, or `PreTest` connections, depending on the application server.

With this approach, when a FAN event is received during planned maintenance, Fast Connection Failover closes sessions when they are tested, because the application does not have a connection to the database at this time, and can retry for a new connection. The connection tests may use `isValid`, `isClosed`, `isUsable`, `PingDatabase`, or a SQL statement that is preceded by the hint `/*+ CLIENT_CONNECTION_VALIDATION */`.

- For a SQL test, the SQL syntax must begin with the hint `/*+ CLIENT_CONNECTION_VALIDATION */`. At the time the SQL command runs, the driver will drain the connection, if it is affected by the upcoming planned maintenance. Connection pools, data sources, and, in the programmatic case, customer applications, must all be ready to manage the recoverable error that occurs when the SQL command runs, which usually closes the physical connection.

 **Note:**

The SQL hint must be placed as the first non-comment token within the SQL string to avoid changing current, driver-based SQL parsing. For example:

```
/*+ CLIENT_CONNECTION_VALIDATION */ SELECT 1 FROM DUAL;
```

- Third-party Java application servers and Java applications can use the `PooledConnection` standard interface when developing connection pools.
- Beginning with the 11.2.0.3 release of the OCI/OCCI driver, when the `OCI_ATTR_SERVER_STATUS` server context handle attribute returns `OCI_SERVER_NOT_CONNECTED`, the application must terminate the connection. Work will be drained for planned maintenance. The 12.2 release of the driver can also detect `OCISessionRelease` and `OCIRequestEnd` when it receives a planned DOWN event.

FAN callouts are server-side scripts or executables that run whenever a FAN event is generated. You can design and build callouts to do many things. For example:

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

FAN events are published using Oracle Notification Service and Oracle Streams Advanced Queuing, the latter being continued for backward compatibility to previous Oracle Database releases. The publication mechanisms are automatically configured as part of your Oracle RAC installation. If you are using thin Java JDBC connections,

then the client can be automatically configured for Oracle Notification Service, by obtaining the database-server Oracle Notification Service configuration from the database connection. It is not necessary to configure Oracle Notification Service on the client.

Oracle Net Services listeners and Global Data Services (GDS) are integrated with FAN events, enabling the listener and GDS to immediately de-register services provided by the failed instance and to avoid erroneously sending connection requests to failed instances.

If you specify the connection load balancing goal `CLB_GOAL_SHORT` for the service, 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.

Fast Application Notification High Availability Events

This section describes the information delivered in the FAN event to a callout program.

FAN event types are listed following the example, and [Table 6-1](#) describes name-value pairs for the event parameters. The event type is always the first entry when you receive FAN information through a callout, as in the following example:

```
SERVICEMEMBER VERSION=1.0
  service=test.company.com database=ractest
  instance=ractest11 host=ractest1_host0343_1 status=up reason=FAILURE
  timestamp=2018-05-08 22:06:02 timezone=-07:00 db_domain=company.com
```

Note that the preceding example displays as one line.

FAN event types include:

```
DATABASE
INSTANCE
NODE
SERVICE
SERVICEMEMBER
SERVICEMETRICS
```

The `DATABASE` and `INSTANCE` types list the default database service as `DB_UNIQUE_NAME`.

All events except for `NODE` events include a `db_domain` field.

Events of `SERVICEMETRICS` type are load balancing advisory events.

See Also: [Table 5-1](#) for more information about load balancing events

Table 6-1 Event Parameter Name-Value Pairs and Descriptions

Parameter	Description
VERSION	Version of the event record. Used to identify release changes.
database	The unique name of the database supporting the service; matches the initialization parameter value for <code>DB_UNIQUE_NAME</code> , which defaults to the value of the <code>DB_NAME</code> initialization parameter.
instance	The name of the instance that supports the service; matches the <code>ORACLE_SID</code> value.

Table 6-1 (Cont.) Event Parameter Name-Value Pairs and Descriptions

Parameter	Description
host	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	<p>The service name; matches the name of the service as listed in DBA_SERVICES and is domain-qualified as appropriate. Refer to the following examples:</p> <pre>SERVICEMEMBER VERSION=1.0 service=swingbench database=orcl instance=orcl_2 host=rwsbj13 status=up reason=USER card=1 timestamp=2018-05-29 17:26:37 timezone=-07:00 db_domain=</pre> <pre>SERVICEMEMBER VERSION=1.0 service=swingbench.example.com database=orcl instance=orcl1 host=rwsbj09 status=up reason=USER card=2 timestamp=2018-05-03 17:29:28 timezone=-07:00 db_domain=example.com</pre> <pre>SERVICEMEMBER VERSION=1.0 service=swingbench.example.com database=orcl instance=orcl2 host=rwsbj10 status=up reason=USER card=1 timestamp=2018-07-03 17:29:18 timezone=-07:00 db_domain=example.com</pre>
status	<p>Values are UP, DOWN, NODEDOWN (this only applies to the NODE event type), NOT_RESTARTING, and UNKNOWN.</p> <p>Notes:</p> <ul style="list-style-type: none"> When the node is down, the status is NODEDOWN, as opposed to DOWN for other event types. When STATUS=NODEDOWN and REASON=MEMBER_LEAVE, a node has failed and is no longer part of the cluster, or a user has stopped a node. When STATUS=NODEDOWN and REASON=PUBLIC_NW_DOWN, the node is up but it is unreachable because the public network is down because of either a failure or a user action. Multiple public networks are supported by Oracle Clusterware. The FAN event reflects this fact.
reason	<p>AUTOSTART, BOOT, DEPENDENCY, FAILURE, MEMBER_LEAVE, PUBLIC_NW_DOWN, USER.</p> <p>Notes:</p> <ul style="list-style-type: none"> For DATABASE and SERVICE event types, REASON=AUTOSTART if, when the node started, the AUTO_START resource attribute was set to restore, and the resource was offline before the node started. For DATABASE and SERVICE event types, REASON=BOOT if, when the node started, the resource started because it was online before the node started. For SRVCTL and Oracle Enterprise Manager operations, REASON=USER describes planned actions for such operations as draining work.

Table 6-1 (Cont.) Event Parameter Name-Value Pairs and Descriptions

Parameter	Description
cardinality	The number of service members that are currently active; included in all <code>SERVICEMEMBER UP</code> events. Following is an example of <code>SERVICEMEMBER UP</code> event: <pre>SERVICEMEMBER VERSION=1.0 service=swingbench.example.com database=orcl instance=orcl_2 host=mjkbj09 status=up reason=USER card=1 timestamp=2018-07-12 14:46:46 timezone=-07:00 db_domain=example.com</pre>
incarnation	For <code>NODEDOWN</code> events; the new cluster incarnation. This value changes each time a member joins or leaves the cluster. Following is an example of a <code>NODEDOWN</code> event: <pre>NODE VERSION=1.0 host=stru09 incarn=175615351 status=down reason=member_leave timestamp=27-Jul-2018 14:49:32 timezone=-07:00</pre>
timestamp	The time according to Oracle Clusterware that an event occurs.
timezone	The time zone of Oracle Clusterware where the event occurred, given as GMT +/-hh:mm.

Some of the FAN event record parameters have values that correspond to values returned by the `SYS_CONTEXT` function using the default namespace `USERENV`, as shown in [Table 6-2](#).

Table 6-2 FAN Parameters and Matching Session Information

FAN Parameter	Matching Session Information
<code>SERVICE</code>	<code>sys_context('userenv', 'service_name')</code>
<code>DATABASE_UNIQUE_NAME</code>	<code>sys_context('userenv', 'db_unique_name')</code>
<code>INSTANCE</code>	<code>sys_context('userenv', 'instance_name')</code>
<code>CLUSTER_NODE_NAME</code>	<code>sys_context('userenv', 'server_host')</code>

Subscription to High Availability Events

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. If you are using an OCI client or an ODP.NET client to receive FAN events, then you must enable the service used by that client to access the alert notification queue by using `SRVCTL` with the `-notification` parameter.

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 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 `Grid_home/racg/usrco` directory on every node that runs Oracle Clusterware. The executable must be able to run standalone when called, with optional arguments, from another program. The following is an example of an executable shell script, named `callout.sh`, which is placed in the `Grid_home/racg/usrco` directory:

```
#!/bin/bash
FAN_LOGFILE= [your_path_name]/admin/log/'hostname'_uptime'.log
echo $* "reported="'date' >> $FAN_LOGFILE &
```

The previous example adds entries similar to the following in the log file, indicated by `$FAN_LOGFILE` in the shell script, each time a FAN event is generated:

```
NODE VERSION=1.0 host=sun880-2 incarn=23 status=nodedown reason=public_nw_down
timestamp=08-Oct-2012 04:02:14 timezone=-08:00 reported=Fri Oct 8 04:02:14 PDT 2012
```

The contents of a FAN event record matches the current session of the user logged on to the database, as shown in [Table 6-2](#). The user environment (`USERENV`) information is also available using OCI connection handle and descriptor attributes (using `OCIAttrGet()`). Use this information to take actions on sessions that match the FAN event data.



See Also:

[Table 6-1](#) for information about the callout and event details

In general, events are only posted to user callouts on the node from which the event originated. For example, if the database on `node1` goes down, then the callout is posted to `node1`, only. The only exceptions to this are node down and VIP down events—these events are posted to all nodes, regardless of from where they originated.

Managing Unplanned Outages

You can assign services to one or more instances in an administrator-managed Oracle RAC database or to server pools in a policy-managed database.

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 attempts to maintain the cardinality of the service. If the service definition allows for failover and that is necessary to maintain cardinality, then failover occurs.

FAN events can occur at various levels within the Oracle Database architecture and are published through Oracle Notification Service and Oracle Streams Advanced

Queuing for backward compatibility with previous OCI clients. FAN callouts can also be written to execute on the database server in response to FAN events.

**Note:**

Oracle Database does not run Oracle RAC callouts with guaranteed ordering. Callouts are run asynchronously and they are subject to scheduling variability.

FAN is published 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 Oracle Automatic Storage Management (Oracle ASM) processes, not just the database. You can use FAN callouts to report faults to your fault management system and to initiate repair jobs.

It is a complex task for application developers to mask outages of a database session (instance, node, storage or network, or any other related component) and, as a result, errors and timeouts are often exposed to the end users leading to user frustration, lost productivity, and lost opportunities. Together, FAN and Application Continuity mask outages from users and applications by recovering the in-flight work for impacted database sessions following outages. Application Continuity performs this recovery beneath the application so that the outage appears to the application as a slightly delayed execution of the request.

Managing Planned Maintenance

For repairs, upgrades, and changes that require you to isolate one or more instances or nodes, Oracle RAC provides interfaces that relocate, disable, and enable services to minimize service disruption to application users.

When you relocate a service, you indicate the service should run on another instance temporarily. If you forcibly disable a service, then the service is stopped on all database instances and is no longer available. Disabled services are not restarted automatically. When a service is stopped or relocated, FAN is published with a planned reason code, typically `reason=user`. Once you complete the operation, you can return the service to normal operation or enable the service and then restart it. When a service restarts, FAN is published with `UP` status codes.

Due to dependencies, if you manually shut down your database, then all of your services for that database automatically stop. If you want your services to automatically start when you manually restart the database, then you must set the management policy of the service to automatic. If you want to shut down only one instance of the database and you want the service to remain offered, then you can either relocate the service using `srvctl relocate service` or stop the instance using `srvctl stop instance`, which enables the service to automatically failover according to its failover policy.

In either case, Oracle recommends that work running under the service drain at request boundaries. The drain interval is specified as an attribute of the service or you can provide a drain interval on the SRVCTL command line.

This section includes the following topics:

- [Managing Planned Maintenance Without User Interruption](#)

- [Managing a Group of Services for Maintenance](#)
- [Server Draining Ahead of Planned Maintenance](#)

Managing Planned Maintenance Without User Interruption

Oracle recommends that you drain database sessions from the instance over a controlled time period from FAN-enabled Oracle or non-Oracle connection pools, or, beginning with Oracle Database 18c at the database, itself.

Draining database sessions is the safest way to migrate work without interrupting applications. When draining occurs at connection tests and outside of request boundaries, it is 100% correct. Applications continue with no interruption as existing work completes and new work acquires a session for the same service functioning at another instance, resulting in no errors returned to applications and no risk of incorrect database session states. For connection tests, the caller expects to receive a good or bad return code and is ready to handle the result, making inspecting connection tests a widely applicable and very powerful solution.

The service attributes `-drain_timeout` and `-stopoption` control the drain time period, and then how the service manages sessions that have not completed once this time period expires. Requests that complete and then check back in to the pool or close, can be directed to a new location that is not affected by the planned maintenance.

Application Continuity provides additional cover, giving continuous service for those requests that do not complete within the allotted drain time.

Using any FAN-aware pool with Fast Connection Failover configured (such as OCI session pools, Oracle Universal Connection Pool, Oracle WebLogic Server Active GridLink for Oracle RAC, or ODP.NET) allows sessions to drain at [request](#) boundaries after receipt of the FAN planned `DOWN` event.

Because not every application uses an Oracle connection pool and not every application is FAN-aware, beginning with Oracle Database 18c, the database inspects sessions during planned maintenance seeking safe places to stop a session so that the application is not disrupted. After stopping the service, the database looks for safe places where the connection can be closed. When the connection is closed, the database cleans up the session.

Stopping a session at a safe place enables the application to open a new connection with the states that it needs. Draining sessions may take a period of work to flow through each session. There is no requirement that closing a session is immediate, rather the close must occur at a safe place that exposes no errors to the application, and, preferably, before the drain timeout period has expired.

Requests are far more important than transactions because they enable the issued work to complete. For draining requests, the Oracle Universal Connection Pool uses the drain timeout to gradually drain, which prevents an overload of logins on the instances drained, by slowly releasing the original sessions across the time period rather than all at once. Gradual draining has the benefit of not disturbing the other work ongoing at the target instances.

Both `DRAIN_TIMEOUT` and `STOP_OPTION` are service attributes that you can define when you add the service or modify it after creation. You can also specify these attributes using `SRVCTL`, which will take precedence over what is defined on the service. You can specify the `-drain_timeout` and `-stopoption` parameters when using the following `SRVCTL` commands:

- `srvctl add service`
- `srvctl modify service`
- `srvctl relocate service`
- `srvctl stop service`
- `srvctl stop database`
- `srvctl stop instance`

To manage planned maintenance without user interruption:

1. Use SRVCTL to relocate a singleton service or a service not running on all nodes. Use the `-force` flag with the previously listed SRVCTL commands. You must use the `-force` flag if you specify the `-stopoption` parameter on the command line when you run either `srvctl relocate service` or `srvctl stop service`. For example:

```
$ srvctl relocate service -database mycdb01 -service myservice -drain_timeout 120  
-stopoption IMMEDIATE -oldinst mycdb01_01 -force
```

The preceding command relocates the service named `myservice01` from the instance named `mycdb01_01` to any instance on which it is configured to run. Oracle Clusterware chooses this instance if you do not specify a target on the command line, and waits two minutes (in this example) for any active sessions to drain, after which any sessions remaining on `mycdb01_01` are forcibly disconnected. The connection pool automatically releases a connection at a request boundary.

 **Note:**

If the service you want to relocate is a uniform service that is currently running on all nodes, then the preceding command returns an error, unless the service is not up on all instances, in which case the preceding command example would succeed for a uniform service.

2. The FAN planned `DOWN` event clears idle sessions from the connection pool immediately and marks active sessions to be released at the next check-in. These FAN actions drain the sessions from the instance without disrupting the users.

Existing connections on other instances remain usable, and new connections can be opened to these instances if needed.

3. Not all sessions, in all cases, will check their connections into the pool. Oracle recommends, as a best practice, to have a timeout period (by setting the `-drain_timeout` parameter), after which the instance is forcibly shut down or the service stopped, evicting any remaining client connections.

After the drain interval expires, the `-stopoption` parameter is implemented, which you can define against a service or a database, as follows:

- When stopping a service (`srvctl stop service`), you can specify one of the following stop options using the `-stopoption` parameter: `TRANSACTIONAL` or `IMMEDIATE`
- When stopping a database (`srvctl stop database`), you can specify one of the following stop options using the `-stopoption` parameter: `NORMAL`, `TRANSACTIONAL`, `IMMEDIATE`, or `ABORT`

The database stop options correlate to the service stop options, as follows:

```
NORMAL=NONE  
TRANSACTIONAL/TRANSACTIONAL LOCAL=TRANSACTIONAL  
IMMEDIATE/ABORT=IMMEDIATE
```

For those services that are configured to use Application Continuity, an attempt is made to recover these remaining sessions, after they are terminated, masking the outage from users and applications.

If you do not want Application Continuity to attempt to replay sessions, then specify the `-noreplay` flag on the SRVCTL command line.

4. Once maintenance is complete, restart the instance and the service on the original node.
5. The FAN UP event for the service informs the connection pool that a new instance is available for use, allowing sessions to be created on this instance at next request boundaries.

Managing a Group of Services for Maintenance

Many enterprises run a large number of services, whether it be many services offered by a single database or instance, or many databases offering a few services running on the same node.

You no longer need to run SRVCTL commands for each individual service but need only specify the node name, database name, pluggable database name, or the instance name for all affected services.

- For example, if you want to stop all of the services running on a given node, then you could use the following command:

```
$ srvctl stop service -node racnode01 -drain_timeout 60 -stopoption IMMEDIATE
```

The command stops all services running on `racnode01`, allowing a drain interval of 60 seconds. After 60 seconds any remaining sessions are stopped immediately. The 60-second drain timeout interval overrides any attribute setting on any of the services.

The command could also be qualified to stop the databases on a node, as in the following example:

```
$ srvctl stop instance      -node racnode01 -drain_timeout 60 -stopoption  
TRANSACTIONAL  
    LOCAL -failover -force
```

When you specify the `-failover` parameter:

- All services are relocated, if possible, respecting the drain timeout interval and the stop option specified.
- Any services that cannot be failed over are stopped, using the stop option specified.
- Wait for the length of the drain timeout interval or until all sessions for targeted services are removed, whichever is sooner.
- All instances stop according to the stop option specified.

When you specify the `-stopoption TRANSACTIONAL LOCAL` parameter:

- Remaining services stop according to the drain timeout interval and stop option specified.
- Wait for the length of the drain timeout interval or until all sessions for targeted services are removed, whichever is sooner.
- The instance stops using the `TRANSACTIONAL LOCAL` stop option.

This section includes the following topics:

- [Starting Services](#)
- [Pluggable Database-Level Operations](#)
- [Relocating Services](#)
- [Stopping Services](#)

Starting Services

You can use the `srvctl start service` command to start all services on a node, all services offered by a database, all services offered by a pluggable database, or all services offered on an instance or within a given server pool.

You can also supply a list of services (a subset of all services) to the `srvctl start service` command that you want to start. Additionally, you can provide a node restriction, used in conjunction with the database option, for all services that can be started on a particular node. You can restrict the `srvctl start service` command to start only the parallel query service by specifying the `-pq` parameter.

The following examples illustrate how you can start services:

- To start all of the services offered by a single pluggable database:

```
$ srvctl start service -db myRACDB01 -pdb myPDB01 -startoption OPEN
```

To start all services on a given database and any of its pluggable databases:

```
$ srvctl start service -db myRACDB
```

To start a list of services on a given database, regardless of any pluggable database with which they are associated:

```
$ srvctl start service -db myRACDB -service
"myFirstService,mySecondService,myThirdService"
```

To start all services for a database that can run on a given node:

```
$ srvctl start service -d myRACDB -node racnode01
```

Pluggable Database-Level Operations

You can use SRVCTL to manage services on pluggable databases.

- To start all services for a pluggable database, for all instances or a single instance:

```
$ srvctl start service -db db_name -pdb pdb_name [-instance instance_name]
```

- To stop all services for a pluggable database, for all instances or a single instance:

```
$ srvctl stop service -db db_name -pdb pdb_name [-node node_name | -instance
inst_name | -serverpool pool_name] [-stopoption stop_option] [-drain_timeout
timeout]
[-force [-noreplay]]
```

 **Note:**

The `-pdb pdb_name` parameter is optional. If you omit the pluggable database name, then the operation occurs for the entire container database (all pluggable databases within this container).

Relocating Services

You can use the `srvctl relocate service` command to relocate services to a target destination, which can be an instance, a node, or a database.

- In the following command examples, all services are relocated from the named database, pluggable database, instance, or node. The services will only relocate if the target can support that service, as defined by the service configuration. Any services that cannot be relocated remain at the original location. A placement error is recorded against any services that could not be relocated, or were already running at the new target. Services that fail to relocate remain running at their original location, and any sessions remain active.

```
$ srvctl relocate service -database myRACDB -oldinst RACDB_01 -newinst  
RACDB_03  
-drain_timeout 30 -force
```

or

```
$ srvctl relocate service -database myRACDB -pdb myPDB01 -currentnode racnode01  
-targetnode racnode02 -drain_timeout 30 -force -noreplay
```

The relocate operation starts the service in the new location prior to stopping the service in its existing location.

- If you do not specify a target destination, then Oracle Clusterware relocates all services or specific services from the specified database, pluggable database, instance, or node, as in the following examples:

```
$ srvctl relocate service -database myRACDB -service "myService01,myService02"  
-drain_timeout 30 -force
```

or

```
$ srvctl relocate service -database myRACDB -pdb myPDB01 -drain_timeout 30  
-force -noreplay
```

If there is no valid target available, then the service remains at the original location and the sessions remain active. You must examine the services and stop them if that is what you want to do.

When you relocate a service, it starts at the new location before it stops at the original location. Oracle Clusterware can start that new instance or pluggable database as a dependency. When specified, the `-drain_timeout` and `-stopoption` parameters override the service attributes.

Stopping Services

You can use the `srvctl stop service` command to stop all services on a node, all services offered by a database, all services offered by a pluggable database, or all services offered on an instance or within a given server pool.

You can also supply a list of services (a subset of all services) that you want to stop to the `srvctl stop service` command, and you can also restrict the `srvctl stop service` command to stop only the parallel query service by specifying the `-pq` parameter.

The following examples illustrate how you can stop services:

- To stop all of the services offered by a single pluggable database:

```
$ srvctl stop service -db myRACDDB01 -pdb myPDB01 -drain_timeout 15 -stopoption TRANSACTIONAL
```

To stop all services on a given database and any of its pluggable databases:

```
$ srvctl stop service -db myRACDB -drain_timeout 15 -stopoption IMMEDIATE
```

To stop only a subset of the services offered by a database:

```
$ srvctl stop service -db myRACDB -service "myFirstService,mySecondService, myThirdService" -drain_timeout 60 -stopoption IMMEDIATE
```

 **Note:**

If you use the `-wait YES` SRVCTL command line parameter, then the `-stopoption` parameter is not enforced until the entire drain timeout interval has expired, even if all of the sessions have exited prior to this interval completing.

Server Draining Ahead of Planned Maintenance

Before planned maintenance, drain or fail over database sessions at the database instance so application work is not interrupted. Beginning with Oracle Database 18c, the database itself drains the sessions.

When you prepare for planned maintenance, you must stop or relocate the services that are using the server infrastructure. Relocating services is done over a period of time prior to the planned outage and is based on the nature of work associated with each service.

Moving or stopping a service triggers a FAN notification that is received by the subscribing Oracle drivers and Oracle connection pools. Immediately, new work to that service is directed to another functioning instance of that service. Existing sessions are marked for release after their work completes. As work completes and the connections are returned to the connection pool, either the Oracle driver or the connection pool terminates these sessions.

For third-party OLTP applications, application servers, and custom applications, which all have their own connection pools that borrow and return database sessions, it is safe to drain a database session when it is no longer borrowed. The optimal point for the Oracle server infrastructure to close a session is when the application server tests the validity of that connection. No error is returned to the application when the

connection pool manager tests the validity of connections when borrowing and releasing, and also finds that the connection is no longer valid.

Not all sessions can be drained. If Transparent Application Continuity or Application Continuity is enabled, then the server detects request boundaries where Application Continuity can recover the session fast. The server can interrupt the session, which Application Continuity recovers elsewhere with no interruption. Transparent Application Continuity does not require that you configure initial state, nor that you disable side effects.

The procedure for rolling planned maintenance moves services in advance of maintenance to another database instance, and notifies the client-side drivers, connections pools, the database instance itself, and other subscribers that maintenance is pending, and what needs to be drained (either connections or sessions using this service). Once notified of draining, a FAN event is sent and the client pools behave as described elsewhere, in addition, the database begins to search for safe places to release connections and, if needed, to migrate the connections.

A safe place is a point where an application is not disturbed. In the case of connection pools, that means connections that are not borrowed (checked-in), and, in the case of applications, the same applies at the point of borrowing or returning a connection. At this time, all work is either complete or not started. The database can also fail over connections when all states can be restored transparent to the application.

The database uses an extensible set of rules and heuristics to detect when to take the database session away. When draining starts, the database session persists at the database until a rule is satisfied. The rules include the following:

- Standard application server tests for validity
- Custom SQL tests for validity
- Request boundaries are in effect and no request is active
- Request boundaries are in effect and the current request has ended
- The session has one or more session states that are recoverable, and can be recreated at failover

 **Note:**

To drain connections beginning with Oracle Database 18c, see ["Adding, Disabling, Enabling, and Removing Connection Tests for Draining at the Server"](#).

In the case of connection tests, for example, it is standard practice for application servers, pooled applications, job schedulers, and others, to test connections when borrowed from connection pools, when returned to the pool, and at batch commits. When draining, the database intercepts the connection test, closes the connection and returns a failed status for the test. The application layer issuing the connection test is ready to handle a failed return status and, typically, issues a further request, to obtain a different connection. The application is not disturbed.

Some database sessions do not drain. For these sessions, the database must find a break point when the session can be replaced. At a break point a connection can be failed over transparently when states are known and recoverable. Break points can be transaction boundaries, a request starting (beginRequest), before calls are executed in

that request, and patterns, such as an audit call that signals that a request is starting or ending. Break points apply only when states are known to be restorable.

Failing over connections requires that you enable Application Continuity, Transparent Application Continuity, or transparent application failover (TAF), depending on your application.

**Note:**

Oracle recommends that you use Oracle connection pools such as UCP or OCI Session Pool because these offer significant advantages in providing continuous availability, load balancing, and so on.

Adding, Disabling, Enabling, and Removing Connection Tests for Draining at the Server

You can add a SQL connection test to a service, a pluggable database, or non-container database.

There are four SQL connection tests added for every database service and pluggable database service, by default, so, if an application uses these following SQL connection tests on the connection, then you do not need to add them:

```
SELECT 1 FROM DUAL;  
SELECT COUNT(*) FROM DUAL;  
SELECT 1;  
BEGIN NULL;END;
```

- To add a server-side SQL connection test for a service, use a SQL statement similar to the following:

```
SQL> execute dbms_session.add_sql_connection_test('select dummy from  
dual', 'sw_orcl');
```

To add a server-side SQL connection test for a pluggable database or non-container database, log on to the non-container database and use a SQL statement similar to the following:

```
SQL> execute dbms_session.add_sql_connection_test('begin null;end;');
```

After you add the SQL connection test, it will be enabled by default.

- You can disable a SQL connection test if you do not need it or it is not in use by logging on to a pluggable database or non-container database and using a SQL statement similar to the following:

```
SQL> execute dbms_session.disable_connection_test(dbms_session.sql_test, 'select  
dummy from dual');
```

By default, the ping test and end request test are disabled but, if you want to disable them after enabling them, then you can use either of the following SQL statements:

If you want to disable the ping test, then use a SQL statement similar to the following:

```
SQL> execute dbms_session.disable_connection_test(dbms_session.ping_test);
```

If you want to disable the end request test, then use a SQL statement similar to the following:

```
SQL> execute dbms_session.disable_connection_test(dbms_session.endrequest_test);
```

- You can enable a SQL connection test after you disable it by logging on to the pluggable database or non-container database and using a SQL statement similar to the following:

```
SQL> execute dbms_session.enable_connection_test(dbms_session.sql_test,'select dummy from dual');
```

You can also enable the ping test and end request test if they are disabled by using either of the following SQL statements:

If you want to run any test that uses ping such as `isValid`, `isUsable`, `OCIping`, or `connection.status`, then use a SQL statement similar to the following:

```
SQL> execute dbms_session.enable_connection_test(dbms_session.ping_test);
```

If you want to enable draining at the end of a request, then use a SQL statement similar to the following:

```
SQL> execute dbms_session.enable_connection_test(dbms_session.endrequest_test);
```

If you want to disable draining on the end of a request, then use a SQL statement similar to the following:

```
SQL> execute dbms_session.disable_connection_test(dbms_session.endrequest_test);
```

- You can remove a SQL connection test if it is not needed by logging on to the pluggable database or non-container database and running SQL statements similar to the following:

```
SQL> execute dbms_session.delete_sql_connection_test('select dummy from dual','sw_orcl');
SQL> execute dbms_session.delete_sql_connection_test('begin null;end;');
```

Every application server has a feature to test the validity of the connections in their respective connection pools, which is set either by a configuration property or at the administrative console. The purpose of the test is to prevent vending an unusable connection to an application, and when an unusable connection is detected, to remove it when released to the pool.

Across the various application servers, the tests have similar names. The tests offered use various approaches, the most common being a SQL statement. Oracle recommends that Java application servers use the standard Java call `connection.isValid`. Beginning with Oracle Database 18c, these tests are used to drain the database. Also beginning with Oracle Database 18c, the database drains sessions without using FAN by inspecting sessions for safe draining points.

The following table describes the standard connection tests available for several of the more common application servers:

Table 6-3 Standard Connection Tests for Some Common Application Servers

Application Server	Connection Test to Database
Oracle WebLogic Server	<p>The tests offered include:</p> <ul style="list-style-type: none"> • <code>dbms_session.enable_connection_test(dbms_session.sql_test,'select 1 from dual');</code> • <code>TestConnectionsOnReserve:</code> <code>isUsable, isValid, Or PingDatabase</code> • <code>TestConnectionsOnCreate (SQL syntax) for server draining:</code> <code>Select 1 from dual;</code>
Oracle WebLogic Server Active Gridlink	<p>The test is embedded: <code>isUsable</code></p>
IBM WebSphere	<p><code>dbms_session.enable_connection_test(dbms_session.sql_test,'select 1 from dual');</code></p> <p><code>TestConnectionsOnCreate (SQL syntax) for server draining:</code> <code>Select 1 from dual;</code></p>
RedHat JBoss	<p><code>check-valid-connection-sql (SQL syntax):</code> <code>Select 1 from dual;</code></p>
Apache Tomcat	<p>There are two tests available—<code>testOnBorrow</code> and <code>testOnReturn</code>—and they both use SQL syntax to test the connection to the database:</p> <p><code>dbms_session.enable_connection_test(dbms_session.sql_test,'select 1 from dual');</code></p> <p>Application server uses: <code>Select 1 from dual;</code></p>

Oracle recommends that you use the following format for supporting automatic configuration of Oracle Notification Services (ONS), so that you can receive FAN events (over ONS):

Example 6-1 Automatic Configuration of FAN

```
alias =(DESCRIPTION =
  (CONNECT_TIMEOUT=90)(RETRY_COUNT=20)(RETRY_DELAY=3)(TRANSPORT_CONNECT_TIMEOUT=3)
  (ADDRESS_LIST =
    (LOAD_BALANCE=on)
    ( ADDRESS = (PROTOCOL = TCP)(HOST=primary-scan)(PORT=1521)))
  (ADDRESS_LIST =
    (LOAD_BALANCE=on)
    ( ADDRESS = (PROTOCOL = TCP)(HOST=secondary-scan)(PORT=1521)))
  (CONNECT_DATA=(SERVICE_NAME = gold-cloud)))
```

About Application Continuity

Application Continuity masks many recoverable database outages (when replay is successful) from applications and users by restoring the database session: the full session, including all states, cursors, variables, and the last transaction if there is one.

Application Continuity addresses the problem that arises when an application is trying to access the database and the database instance becomes unavailable due to an unplanned or planned outage (timeout, network outage, instance failure, repair, configuration change, patch apply, and so on). Without Application Continuity in place, database recovery does not mask outages to applications and end users. In such scenarios, developers and users must cope with exception conditions, and users can be left not knowing what happened to their funds transfers, time sheets, orders, bill payments, and so on. Users might lose screens of uncommitted data, and must log in again and reenter that data. In the worst cases, the administrator might be forced to restart the middle tier to recover from an overwhelming number of logins.

With Application Continuity, if the database instance became unavailable, then Application Continuity attempts to rebuild the session and any open transactions using the correct states; and if the transaction committed and need not be resubmitted, then the successful return status is returned to the application. If replay is successful, then the request can continue safely without risk of duplication. If replay cannot restore data that the application has already processed and potentially made decisions on, then the database rejects the replay and the application receives the original error.

Application Continuity performs the recovery of in-flight transactions and database session state, while ensuring the transaction idempotence provided by Transaction Guard. Each database session is tagged with a logical transaction ID (LTXID), so the database recognizes whether each replay committed any transactions, and if it did commit any transactions, whether the work also ran to completion. While Application Continuity attempts to replay, the replay appears to the application as a delayed execution, or the application receives the commit response for the original transaction (if the last transaction had completed before the outage).

Application Continuity is supported for Oracle RAC and Oracle Active Data Guard. It is supported for both non-consolidated databases and Multitenant databases (with failover at the pluggable database level). It is not currently supported for Golden Gate, Logical Standby, third-party replication solutions, or DML redirection if using Oracle Active Data Guard.

Key Concepts for Application Continuity

This section describes several terms and concepts that you must understand to use Application Continuity.

The following terms are used throughout this chapter:

Database request

A database request is a unit of work submitted to the database from the application, such as a transaction. A request typically corresponds to the SQL and PL/SQL, and other database calls, of a single web request on a single database connection, and it is generally demarcated by the calls made to check-out and check-in the database connection from a connection pool.

Recoverable error

A recoverable error is an error that arises due to an external system failure, independent of the application session logic that is executing, such as a lost or invalid connection. Recoverable errors occur following planned and unplanned outages of foregrounds, networks, nodes, storage, and databases. The application receives an error code that can leave the application not knowing the status of the last operation submitted. Application Continuity reestablishes database sessions and resubmits the pending work for the class of recoverable errors.

Application Continuity does not resubmit work following call failures due to nonrecoverable errors. An example of a nonrecoverable error that would *not* be replayed is submission of invalid data values.

Commit outcome

A transaction is committed by updating its entry in the transaction table. Oracle Database generates a redo-log record corresponding to this update and writes out this redo-log record. Once this redo-log record is written out to the redo log on disk, the transaction is considered committed at the database. From the client perspective, the transaction is considered committed when an Oracle message (called the *commit outcome*), generated after that redo is written, is received by the client. However, if a `COMMIT` has been issued, then the `COMMIT` failure message cannot be retrieved if it is not received by the client or the application.

Mutable functions

Mutable functions are non-deterministic functions that can obtain a new value every time they are called, and thus their results can change frequently. Mutable functions cause a problem for replay because the results can change at replay. Consider `sequence.NEXTVAL` and `SYSDATE`, often used in key values. If a primary key is built with values from these function calls, and is used in later foreign keys or other binds, at replay the same function result must be returned.

Application Continuity provides mutable object value replacement at replay for granted Oracle function calls to provide opaque bind-variable consistency. If the call uses database functions that are mutable, including `sequence.NEXTVAL`, `SYSDATE`, `SYSTIMESTAMP`, and `SYSGUID`, the original values returned from the function execution are saved and are reapplied at replay.

Session state consistency

After a `COMMIT` statement has executed, if state was changed in that transaction, it is not possible to replay the transaction to reestablish that state if the session is lost. When configuring Application Continuity, the applications are categorized depending on whether the session state after the initial setup is static or dynamic, and thus whether it is correct to continue past a `COMMIT` operation within a request.

- A session has **dynamic** state if the session state changes are not fully encapsulated by the initialization, and cannot be fully captured by `FAILOVER_RESTORE` or in a callback at failover. After the first transaction completes, failover is internally disabled until the next request begins. Session state may change during the course of the request.
- A session has a **static** state if all session state changes (for example, NLS settings and PL/SQL package state) occur as part of initialization, and can be encapsulated by `FAILOVER_RESTORE` or in a callback at failover. Static applications

are those that were able to use Transparent Application Failover (TAF) before Application Continuity. Session state does not change during the course of the request. (Choose setting session state consistency to `AUTO` over `STATIC` mode, when possible, because auto mode purges and cleans more efficiently than the pre-Application Continuity TAF mode.)

- With Transparent Application Continuity, the state is managed for you by setting session state consistency to `AUTO` (this is a mandatory setting for Transparent Application Continuity). The session states are tracked and managed for you.

Transparent Application Continuity

Applications achieve continuous availability when planned maintenance and unplanned outages of the database are transparent.

Transparent Application Continuity is a functional mode of Application Continuity beginning with Oracle RAC 18c that transparently tracks and records session and transactional state so that a database session can be recovered following recoverable outages. This is done safely and with no need for a DBA to have any knowledge of the application or make application code changes. Transparency is achieved by using a state-tracking infrastructure that categorizes session state usage as an application issues user calls.

You can enable Transparent Application Continuity as default to protect applications during planned maintenance and when unplanned outages occur. For planned maintenance, database sessions that reach a safe place (such as a connection test or a known recoverable point) are drained at the database. For database sessions that do not drain, the database determines where to fail the database session over and invokes Application Continuity to do so. Application Continuity hides unplanned outages for Java-based applications, OCI and ODP.NET applications including SQL*Plus, all Oracle connection pools, Tuxedo, WebLogic Server, and third-party application servers using Universal Connection Pool.

With Transparent Application Continuity, DBAs no longer need to have knowledge of an application to:

- Know that initial state at replay is correct.

At runtime, Transparent Application Continuity records the initial state, if different than before, and records session signatures sufficient to detect deviation in the state of a session at failover. At failover, Transparent Application Continuity restores known session state before replay starts, and verifies that the session state fully matches the original before replay starts, allowing for session state that has been restored using both Application Continuity and other mechanisms such as logon triggers, labels, and connection callbacks. The session state is protected whether one failover or multiple failovers for a request.

- Recognize and disable application-level side effects when recovering a session.

During normal runtime, Transparent Application Continuity detects side effects. The type of side effect is distinguished between those that relate to an application's logic and those that are internal, relating to database housekeeping. For applications that use statements that have side effects, capture is disabled when the statement is running. Once a new request starts, capture is re-enabled automatically.

- Keep mutable values for owned functions.

Mutable functions are functions that can return a new value each time they are run. Oracle provides support for keeping the original results of mutable functions `SYSDATE`, `SYSTIMESTAMP`, `SYS_GUID`, and `sequence.NEXTVAL`. If the original values are not kept and if different values are returned to the application at replay, then Transparent Application Continuity rejects replay. Use grants to keep your sequences, dates, and times. When an application is using its own schema, you can assign the grants for keeping to a role and then grant this role to users.

- For applications using Application Continuity with JDBC thin driver, DBAs do not need to know about request boundaries but when they are in use, Transparent Application Continuity takes advantage of them. It is best practice that you use request boundaries because it is not always possible to identify a checkpoint where a request boundary can be inserted.

Request Boundaries demark where applications and application servers borrow and return connections from their connection pools. Until Oracle RAC 18c, without request boundaries, the lower layers, (such as the database and drivers) have no insight as to how the applications and application servers are managing their connections. Almost every application server and enterprise application, and custom development using good practices, caches their connections in their layer for best performance. The lower layers cannot see how the connections are being handled and balanced. The lower layer could only see user calls to the database.

Using Transparent Application Continuity, the server and the drivers are tracking transaction and session state usage. This allows the driver to detect and inject possible request boundaries. At a possible request boundary, no objects are open, cursors are returned to the driver statement cache, no transactions are open, and the session state is known to be restorable. The driver either closes the current capture and starts new, or enables capture if there had been a disabling event. On the next call to the server, the server verifies and, if applicable, creates a request boundary where there was previously no explicit boundary.

When you use Transparent Application Continuity for Java, your applications will experience lower resource usage and faster recovery because statements that do not contribute to the state are not recorded, or they are purged when no longer needed, and request boundaries are advanced automatically.

For unplanned outages, Transparent Application Continuity is invoked for outages that result in recoverable errors, typically related to underlying software, foreground, hardware, communications, network, or storage layers, hiding most failures from applications and users.

Transparent Application Continuity is enabled when `FAILOVER_TYPE=AUTO`.

Transparent Application Continuity for Various Applications

Transparent Application Continuity covers applications that belong to three different groups, automatically tracked by the state tracking system.

The various applications are of the following types:

- **Request boundaries:** Applications that use containers with request boundaries allow Application Continuity to manage replay between explicit boundaries.
- **Database agnostic:** Applications set a state at connection establishment and do not change non-transactional session states again, or change it rarely. For these applications, Application Continuity identifies implicit boundaries.

- **Black Box:** Applications that are using either Oracle proprietary states, changing states, or both, at run time. This category is further divided:
 - Applications with short user calls, such as OLTP with no visible boundaries
 - Applications with long user calls, such as DSS, reports, and warehouses

Request Boundaries

A [request boundary](#) is a tag that marks the beginning and end of a database request. Beginning with Oracle Database 12c release 2 (12.2.0.1), connection pools that embed request boundaries include Oracle Universal Connection Pool, all WebLogic server data sources, Tuxedo, Oracle Call Interface, ODP.NET Unmanaged Provider, and standard third-party application servers and standalone Java pools that use the Oracle Database 12c JDBC drivers' `PooledConnection` interface, in addition to SQL*PLUS.

When an Oracle database is aware of request boundaries:

- The database can process web requests effectively and with no performance overhead, including when to attach and release connections, so it can multiplex, drain, rebalance, shed, and allow complex states inside requests. Without request boundaries, the lower layers of the database are not aware of web requests. Subsequently, the database relies on Oracle client actions, advisory methods and heuristics, such as fast connection failover, connection validation, and state advice.
- The length of replay is limited to the initial state followed by the user calls in that request less those that are purged by Application Continuity. Request boundaries are an important hint to control the length of replay, and also where to drain for planned maintenance (at the end of the request), and to where to fail over for planned maintenance (at the beginning of the request).
- When using Transparent Application Continuity for Java, request boundaries are implicit and continuously advance. Only the first request boundary is required.
- When using Application Continuity for Java, the replay driver detects safe places to move the request boundaries forward automatically. This feature is only available with AUTO.
- Applications deployed using middle-tier containers that set request boundaries have access to the full set of transparency features that the database server provides. The database detects when a client sets request boundaries and uses the boundaries to mark safe points for draining, failover, concentration, and throughput measures.

Request boundaries enable an application to use all complex, non-transactional session states within a request. The request boundary specification requires that these states are not dependent across boundaries.

Database-agnostic Applications

Database-agnostic applications (applications with no request boundaries) set simple, non-transactional states and use none of the Oracle proprietary features or sequences. These applications often set state once when a connection is created and then do not change state again or change the state infrequently. This category of applications includes those applications that use anonymous PL/SQL that do not create server-side session state.

When using Transparent Application Continuity for JDBC applications, the state classification is used to find a point at which to enable and start recording for Application Continuity after authentication, and to re-enable recording after capture has been disabled by a disabling event. Only the first request boundary is required, and, when request boundaries are there, they are used. Request boundaries are not required for SQL*Plus, and they are embedded for ODP.NET, OCI session pool, Tuxedo, and Oracle Universal Connection Pool.

Application Continuity Operation and Usage

This section explains how Application Continuity works, and how you can use it in applications.

This section includes the following topics:

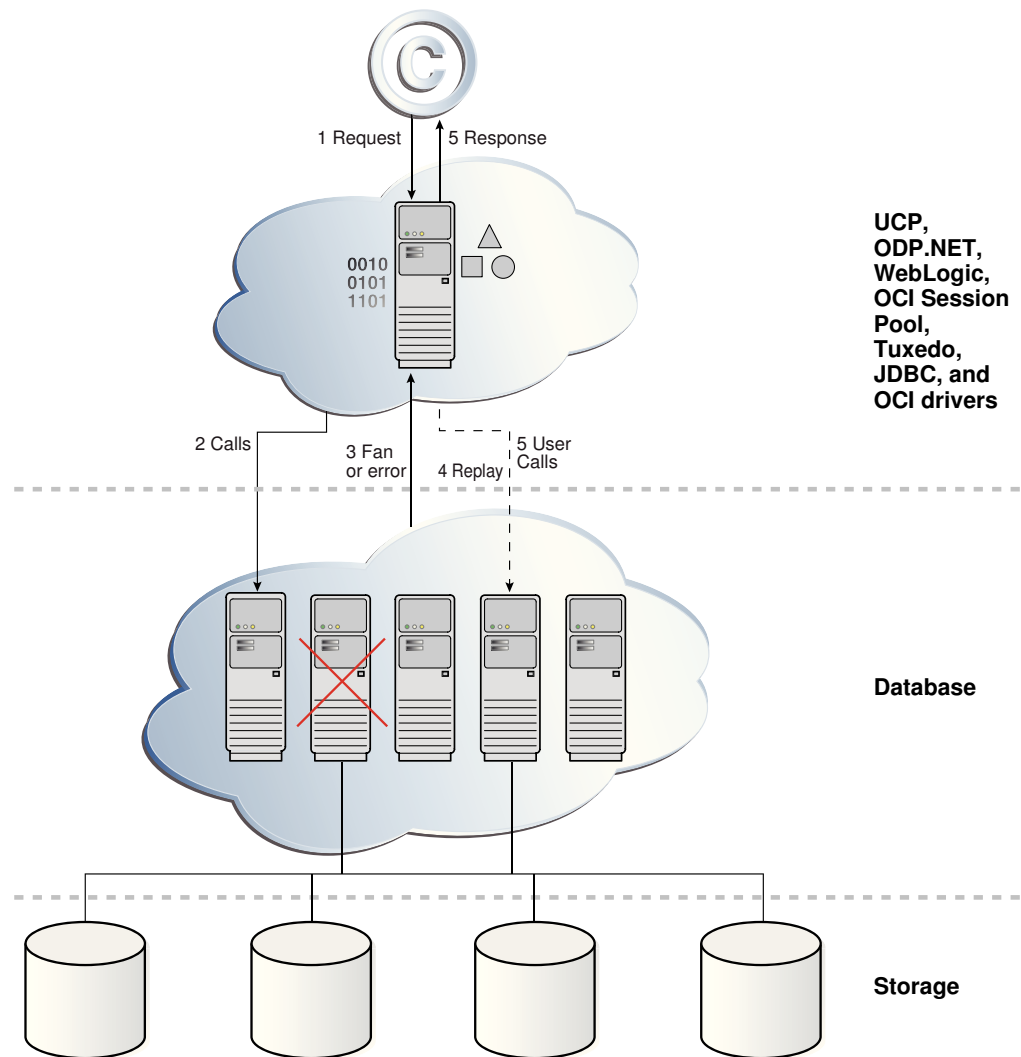
- [How Application Continuity Works for Applications](#)
- [Actions for Using Application Continuity](#)
- [Mutable Functions and Application Continuity](#)
- [Administering Mutable Values](#)
- [Protection-Level Statistics](#)
- [Session State Consistency](#)

How Application Continuity Works for Applications

If a recoverable error occurs and if you enabled replay, then Application Continuity attempts recovery of the database session.

The following figure is a graphical representation of how Application Continuity works.

Figure 6-1 Application Continuity



To attempt to recover a database session following a recoverable error, Application Continuity performs the following steps:

Note:

The steps to recover a database session apply for both unplanned and planned outages, although specific steps vary depending on the type of outage.

1. The client application makes a request, which is passed to a middle tier (such as the Universal Connection Pool (UCP), ODP.NET, WebLogic Server, OCI session pool, Tuxedo, JDBC and OCI drivers, or third-party pool using UCP), or directly to the database using the JDBC replay driver or OCI driver.
2. The JDBC replay driver or OCI driver issues each call in the request.

3. A FAN unplanned or planned down interrupt or recoverable error occurs. FAN/FCF then aborts the dead physical session.
4. Application Continuity begins the replay and does the following:
 - a. Replaces the dead physical session with a new clean session.
 - b. Prepares for replay by using Transaction Guard to determine the outcome of the in-flight transaction, if one was open.
 - c. If `FAILOVER_RESTORE=LEVEL1` or `FAILOVER_TYPE=AUTO`, then Application Continuity restores the common initial session state. Application Continuity uses a label callback or initial callback if an application also sets session states that are not provided by `FAILOVER_RESTORE` in the callback
 - d. Rebuilds the database session, recovering the transactional and non-transactional states, and validating at each step that the data and messages seen by the client driver are the same as those that the client may have seen and used to make a decision.
 - e. Ends the replay and returns to runtime mode.
 - f. Submits the last queued call.

This is the last call made when the outage was discovered. During replay, only this call can execute a `COMMIT`. A `COMMIT` midway through rebuilding the session aborts replay (excluding autonomous transactions).

5. The response is returned to the application.

If replay succeeded, then the application can continue with the problem masked. If not, then the application must handle the original error.

The behavior of Application Continuity after a communication failure depends on the Oracle products and technologies involved. For example:

- If you use Oracle RAC or an Oracle Active Data Guard farm, then, after the connection is reestablished on another running instance, Application Continuity attempts to rebuild the session and replay the last transaction if there is one in flight.
- If you use Oracle Active Data Guard and fail over to a standby site, then Application Continuity connects to the failover instance and attempts to rebuild the session and replay the last transaction there, if a transaction was in-flight. (Application Continuity does not replay if the Oracle Active Data Guard switchover or failover has lost data, and if this is not an Oracle Active Data Guard reader farm with approved lags.)
- If you are using Oracle RAC or Oracle RAC One Node and not using Oracle Active Data Guard, and if an outage causes a break in all public networks or causes the database or database session to shut down briefly, then Application Continuity attempts to rebuild the session and replay the last transaction (if a transaction was in flight) against the database after connectivity is restored.

Actions for Using Application Continuity

Application Continuity masks outages with few or no application changes when you use the Oracle integrated stack.

Application Continuity is available for general use with the following Oracle technologies:

- ODP.NET, Unmanaged Driver
- OCI Session Pool
- Universal Connection Pool
- WebLogic Server
- JDBC Thin Oracle replay driver
- Oracle Tuxedo
- SQL*Plus
- Third-party JDBC application servers with Universal Connection Pool

Application Continuity for Java is embedded in the Universal Connection Pool, WebLogic data sources, including non-XA and XA data sources, and is available with the thin JDBC replay driver, standalone (which is a JDBC replay driver without Oracle connection pools, such as Apache Tomcat or a custom Java connection pool). Application Continuity for OCI is embedded in SQL*Plus, OCI Session Pool. Oracle Tuxedo for non XA data sources, and ODP.NET, Unmanaged Provider.

Application Continuity is embedded in Oracle connection pools. When you use the Oracle connection pools, request boundaries are implicitly marked at check-out and check-in, delimiting the size of each replay. When using third-party connection pools, use UCP if Java, or you can add request boundaries, which are included on the JDK9 standard. When using Transparent Application Continuity for Java, only the first request boundary is required. Request boundaries are discovered using state tracking.

Support for Application Continuity is integrated into many Oracle applications, so the features in such applications are used automatically if you set the Application Continuity-related service attributes. For all applications, follow the steps described in this section.

The main actions for ensuring transparent replay for an application are the following:

1. Only if using Java, determine whether the application uses Oracle JDBC concrete classes. For Application Continuity to be used, the deprecated concrete classes must be replaced.

Use the `-acchk` parameter with the `ORAchk` utility to verify whether an application has any concrete classes. Use a connection without Application Continuity if there is anything that should not be replayed. (Most applications will be replayable.)

See Also:

Oracle Autonomous Health Framework User's Guide for more information about `ORAchk`

2. Ensure that you have the necessary CPU and memory resources.
 - **CPU:** Application Continuity is managed on the client and server sides and requires minimal CPU overhead to operate.

At the client, CPU is used to build proxy objects and for garbage collection (GC).

At the server, CPU is used for validation. CPU overhead is reduced for platforms with current Intel and Sparc chips where validation is assisted in the hardware.

- **Memory:** When using Application Continuity, the replay driver requires more memory than the base driver because the calls are retained until the end of a request. At the end of the request, the calls are released to the garbage collector. This action differs from the base driver that releases closed calls.

The memory consumption of the replay driver depends on the number of calls per request. If this number is small, then the memory consumption of the replay driver is less, and comparable to the base driver.

To obtain the best performance, you must set the same value for both the `-Xmx` and `-Xms` parameters on the client. For example, if there is sufficient memory, then allocate 4 to 8 GB (or more) of memory for the Virtual Machine (VM), for example, by setting `-Xms4g` for 4 GB. If the `-Xms` parameter has a lower value, then the VM also uses a lower value from the operating system, and performance might suffer and garbage collection operations increase.

3. Determine whether the application borrows and returns connections from the connection pool, for example WebLogic Server Pool, Universal Connection Pool, OCI Session Pool, Oracle Tuxedo request, or ODP.NET connection pool, for each request, or whether to add `beginRequest` and `endRequest` APIs to the application's own connection pool to identify request boundaries for Java, only.

! Important:

Do not use the `beginRequest` and `endRequest` Java API calls anywhere other than at request boundaries (borrow and return connections from your connection pool). `endRequest` indicates that the request is complete, and that it is now stateless. Replay starts from the next `beginRequest`. If there is prior state, it must be reestablished using `FAILOVER_RESTORE` or `callback`.

4. Application Continuity replays all states in a request. If the application sets states before vending connections, `FAILOVER_RESTORE` or a `callback` is needed. When using Oracle WebLogic Server or the Universal Connection Pool, use `FAILOVER_RESTORE` or connection labeling. When using OCI session pool, Oracle Tuxedo or ODP.NET with Oracle Database 18c, or later clients, use `FAILOVER_RESTORE`, and only add the TAF `callback` if it is needed. The labeling is used for both runtime and replay.
5. Determine whether the application requires, and therefore needs to configure keeping original values for, `SYSDATE`, `SYSTIMESTAMP`, and `SYS_GUID` and sequences during failover.
6. Assess the application style for the `session_state_consistency` value, and set the appropriate value on the service:
 - If `session_state_consistency` is set to `AUTO`, then Transparent Application Continuity monitors the session state and decides what to do. If you are unsure about state usage or know that states can change in the future, then use Transparent Application Continuity.
 - If `session_state_consistency` is set to `DYNAMIC`, then the application changes the environment or settings during the request. Replay is disabled after the first `COMMIT` until the beginning of the next request. `DYNAMIC` is the default mode, appropriate for most applications.
 - If `session_state_consistency` is set to `STATIC`, then the application *never* changes the session state after initial setup. This mode is typical for database agnostic applications that do not use PL/SQL state and do not use `ALTER part-`

way through transactions. Use this mode with caution, and only for "static" applications.

7. Determine if any requests in the application should not be replayed.

For example, replay may need to be disabled for requests using external PL/SQL actions.

8. Follow these configuration guidelines:

- Use Oracle Database 12c release 1 (12.1.0.1), or later, for Java. Use Oracle Database 12c release 2 (12.2), or later, for OCI-based applications..
- For .NET applications, use ODP.NET, Unmanaged Driver 12.2, or later, connecting to an Oracle Database 12c Release 2 (12.2) or later. By default, Application Continuity is enabled for ODP.NET applications in this configuration.
- For Java-based applications, use Universal Connection Pool 12.1 (or later) or WebLogic Server 12.1.2 (or later) configured with the JDBC Replay data source; or for third party applications, including third party JDBC pools, use JDBC replay driver. For IBM WebSphere, Apache Tomcat, RedHat Spring, and custom Java solutions, the most effective solution is to use UCP as the pooled data source.

Custom Java pools and standalone Java applications can also use the JDBC Replay data source directly. When using custom Java pools and standalone applications, add the `beginRequest` and `endRequest` calls.

- If the application does not borrow and return from the Oracle connection pools, explicitly mark request boundaries. For example, if using custom JDBC pools, or other pools, then call `beginRequest` at check-out and call `endRequest` at check-in. These APIs can also be used for standalone JDBC applications without a connection pool.
- Enable FAN for fast interrupt on errors. This is essential to eliminate a TCP hang occurring before the failover can start. In 12.2 FAN is built into the JDBC and OCI drivers and is on by default for Java.
- Use a database service to connect; never use a SID or an instance name, or the administration service that is the `DB_NAME` or `DB_UNIQUE_NAME`.
- Use a connection string that sets retries for new incoming connections and a delay between these retries.
- For the service, set `FAILOVER_TYPE` to `TRANSACTION` for the manual mode of Application Continuity or set `FAILOVER_TYPE` to `AUTO` for Transparent Application Continuity. Set `COMMIT_OUTCOME` to `TRUE` and, for OCI FAN, set `NOTIFICATION` to `TRUE`. Optionally to find the best connections to use, set `GOAL` to `SERVICE_TIME` and `CLB_GOAL` to `SHORT`.
- Grant permission on the Transaction Guard package, `DBMS_APP_CONT`, to the database users that fail over using Application Continuity, as follows:

```
GRANT EXECUTE ON DBMS_APP_CONT TO user_name;
```

- Use the statistics for request boundaries and protection level to monitor the level of coverage. If you need more details, then use Application Continuity Check Coverage (with the `ORAchk` utility) to report the percentage of requests that are fully protected by Application Continuity, and the location of those requests that are not fully protected. Use this coverage check before deployment and after application changes. Developers and management will

know how well protected an application release is from failures of the underlying infrastructure. If there is a problem, then it can be fixed before the application is released, or waived knowing the level of coverage.

If a connection pool or container does not use an Oracle connection pool, then many third-party Java applications fully support replacing the connection pool with the Universal Connection Pool. This includes IBM WebSphere and Apache Tomcat. Alternatively—for Java applications, only—an application can add its own request boundaries.

See Also:

- *Oracle Data Provider for .NET Developer's Guide for Microsoft Windows* for more information about .NET applications development.
- *Oracle Database JDBC Developer's Guide* for more information about developing applications that include Application Continuity for Java

This section includes the following topics:

- [Configuring Oracle Application Continuity](#)
- [Configuring Oracle Database for Application Continuity](#)
- [Establishing the Initial State Before Application Continuity Replays](#)
- [Delaying the Reconnection in Application Continuity](#)
- [Using Application Continuity with SQL*Plus](#)
- [Using Application Continuity for Planned Maintenance](#)
- [Running Without Application Continuity](#)
- [Disabling Replay in Application Continuity](#)
- [Terminating or Disconnecting a Session Without Replay](#)

Configuring Oracle Application Continuity

If you are using Java, then you must use the `oracle.jdbc.replay.OracleDataSourceImpl`, `oracle.jdbc.replay.OracleConnectionPoolDataSourceImpl`, or `oracle.jdbc.replay.driver.OracleXADataSourceImpl` data source to obtain JDBC connections.

These data sources support all the properties and configuration parameters of all the Oracle JDBC data sources, for example, the `oracle.jdbc.pool.OracleDataSource`.

For OCI based applications including SQL*Plus and ODP.NET, the OCI driver 12.2, and later, supports Application Continuity.

You must remember the following points while using the connection URL:

- If the `REMOTE_LISTENER` setting for the database does not match the addresses in the `ADDRESS_LIST` at the client, then it does not connect, showing `services cannot be found`. So, the `REMOTE_LISTENER` setting for the database *must* match the addresses in the `ADDRESS_LIST` at the client:

- If the connect string uses the SCAN Name, then `REMOTE_LISTENER` must be set to the SCAN name.
- If the connect string uses an `ADDRESS_LIST` of host VIPs, then `REMOTE_LISTENER` must be set to an address list that includes all SCAN VIPs and all host VIPs

 **Note:**

Use SCAN for location independence, so that you do not need to reconfigure the client when you add or delete nodes, or when databases change to running on different nodes.

- Set `RETRY_COUNT`, `RETRY_DELAY`, `CONNECT_TIMEOUT`, and `TRANSPORT_CONNECT_TIMEOUT` parameters in the connection string. These settings improve acquiring new connections at runtime, at replay, and during work drains for planned outages.

The `CONNECT_TIMEOUT` parameter is equivalent to the `SQLNET.OUTBOUND_CONNECT_TIMEOUT` parameter in the `sqlnet.ora` file and applies to the full connection. The `TRANSPORT_CONNECT_TIMEOUT` parameter applies per address.

Set `CONNECT_TIMEOUT` to a high value to prevent an overabundance of log ins. Low values can result in *log in storms* to the application or server pool cancelling and retrying. Do not set $(RETRY_COUNT+1)*RETRY_DELAY$ or `CONNECT_TIMEOUT` larger than your response time SLA. The application must either connect or receive an error within the response time SLA.

These are general recommendations for configuring the connections for high availability. Do not use Easy Connect because this has no high-availability capabilities.

Following is an example of TNS entry. This is the required TNS format for ONS (the transport system for FAN) to be auto configured. Oracle recommends using FAN with Application Continuity to provide fast outage detection.

```
alias =(DESCRIPTION =
  (CONNECT_TIMEOUT=120) (RETRY_COUNT=20) (RETRY_DELAY=3)
  (TRANSPORT_CONNECT_TIMEOUT=3)
  (ADDRESS_LIST =
    (LOAD_BALANCE=on)
    ( ADDRESS = (PROTOCOL = TCP)(HOST=primary-scan)(PORT=1521)))
  (ADDRESS_LIST =
    (LOAD_BALANCE=on)
    ( ADDRESS = (PROTOCOL = TCP)(HOST=secondary-scan)(PORT=1521)))
  (CONNECT_DATA=(SERVICE_NAME = gold-cloud)))
```

Configuring Oracle Database for Application Continuity

Your Oracle Database configuration must include the following to use Application Continuity:

- If you are using Oracle Real Application Clusters (Oracle RAC) or Oracle RAC One Node, Oracle Data Guard, or Oracle Active Data Guard, then ensure that FAN is configured with Oracle Notification System (ONS) to communicate with Oracle Database 12c pools and drivers.
- Set the service attributes on the service for replay and load balancing. For example, set:

- `FAILOVER_TYPE = AUTO` | `TRANSACTION`: Use `FAILOVER_TYPE=AUTO` for Transparent Application Continuity or `FAILOVER_TYPE=TRANSACTION` for manual Application Continuity. This attribute enables the replay functionality for the replay drivers and Application Continuity. Oracle drivers keep track of all replayable statements issued during a database session. If all of the statements are replayable, and any in-flight transactions did not commit or the session is in conversation, then Oracle replays the uncommitted work following a planned or unplanned database outage. This mode re-establishes transactional and non-transaction states automatically with no additional application steps.
- `REPLAY_INITIATION_TIMEOUT = n`: For setting the duration, in seconds, to allow replay to start (where the value of *n* can be, for example, 60, 300, 900, or 1800, depending on your needs)
- `FAILOVER_RETRIES = 30`: For specifying the number of connection retries for each replay
- `FAILOVER_DELAY = 10`: For specifying the delay in seconds between connection retries
- `GOAL = SERVICE_TIME`: If you are using Oracle RAC or Oracle Global Data Services, then this is a recommended setting
- `CLB_GOAL = SHORT`: If you are using Oracle RAC or Oracle Global Data Services, then this is a recommended setting
- `COMMIT_OUTCOME = TRUE`: For using Transaction Guard
- `FAILOVER_RESTORE = AUTO` | `LEVEL1`: Use `FAILOVER_RESTORE=AUTO` for Transparent Application Continuity and `FAILOVER_RESTORE=LEVEL1` for manual Application Continuity. To automatically restore client states that are preset on the connection pool before replay begins—including AUTOCOMMIT state (for Java and SQL*Plus), NLS states, and TAGS (MODULE, ACTION, ECID, CLIENT_ID, CLIENT_INFO) states.
- Grant permission on the Application Continuity package, `DBMS_APP_CONT`, to the database users that fail over using Application Continuity, as follows:


```
GRANT EXECUTE ON DBMS_APP_CONT TO user_name;
```
- *Do not use* the default database service corresponding to the `DB_NAME` or `DB_UNIQUE_NAME`. Also, *do not use* the default database service for high availability, because this service cannot be enabled or disabled, and cannot be relocated on Oracle RAC or switched over to Oracle Data Guard. This service is reserved for Oracle Enterprise Manager Cloud Control (Cloud Control) and for DBAs.

Establishing the Initial State Before Application Continuity Replays

Some applications set an initial state for the connection before allowing applications to use the connection.

Application Continuity must establish this initial state before replay starts. For these applications, `FAILOVER_RESTORE` restores common states listed here. If the states that the application presets are not listed here, and the application needs initial states, then you must add an additional callback.

 **See Also:**

Oracle Database Release Notes for your platform because more parameters are restored in each release

Examples of states that can be preset include:

- PL/SQL package state
- NLS Setting
- Optimizer setting

During a request, Application Continuity reestablishes the entire state for the request. This pre-requisite is for the initial state before Application Continuity starts replaying.

A callback is not required if `FAILOVER_RESTORE` restores all required states, which is the case for most applications.

For applications that set state only at the beginning of a request, or for stateful applications that gain performance benefits from using connections with a preset state, choose one of these options:

- `FAILOVER_RESTORE = LEVEL1` or `AUTO`
- `FAILOVER_RESTORE = NONE` and No Callback
- [Connection Labeling](#)
- [Connection Initialization Callback](#)

FAILOVER_RESTORE = LEVEL1 or AUTO

Setting `FAILOVER_RESTORE` to `LEVEL1` (for manual Application Continuity) or `AUTO` (for Transparent Application Continuity) automatically restores common state initial settings before replaying the request.

Beginning with Oracle Database 18c databases and clients, or later, Oracle recommends setting `FAILOVER_RESTORE` to `LEVEL1` or `AUTO` for all applications.

The states restored include the following:

- `NLS_CALENDAR`
- `NLS_CURRENCY`
- `NLS_DATE_FORMAT`
- `NLS_DATE_LANGUAGE`
- `NLS_DUAL_CURRENCY`
- `NLS_ISO_CURRENCY`
- `NLS_LANGUAGE`
- `NLS_LENGTH_SEMANTICS`
- `NLS_NCHAR_CONV_EXCP`
- `NLS_NUMERIC_CHARACTER`
- `NLS_SORT`

- NLS_TERRITORY
- NLS_TIME_FORMAT
- NLS_TIME_TZ_FORMAT
- TIME_ZONE (OCI, ODP.NET 12201)
- NLS_TIMESTAMP_FORMAT
- NLS_TIMESTAMP_TZ_FORMAT
- CURRENT_SCHEMA
- MODULE
- ACTION
- CLIENT_ID
- ECONTEXT_ID
- ECONTEXT_SEQ
- DB_OP
- AUTOCOMMIT states (for Java and SQL*Plus)
- CONTAINER (PDB) and SERVICE for OCI and ODP.NET

The following are not supported by the THIN driver, so are excluded from the auto-restoration option:

- NLS_COMP
- CALL_COLLECT_TIME
- CLIENT_INFO

FAILOVER_RESTORE = NONE and No Callback

In this scenario, which is applicable to databases and clients prior to Oracle Database 18c, the application does not assume any state when borrowing a connection from a pool, or uses UCP or WebLogic labels to reestablish the initial state.

Beginning with Oracle Database 18c databases and clients, or later, Oracle recommends setting `FAILOVER_RESTORE` to `LEVEL1` or `AUTO` for all applications.

Connection Labeling

Oracle recommends using Connection Labeling, a generic pool feature, as a best practice. When Connection Labeling is present, Application Continuity uses it. Since the Connection Labeling is recreating the state, `FAILOVER_RESTORE` can be set to `NONE`.

This scenario is applicable to Universal Connection Pool (UCP) and Oracle WebLogic server. The application can be modified to take advantage of the preset state on connections. Connection Labeling APIs determine how well a connection matches, and use a callback to populate the gap when a connection is borrowed.

Connection Initialization Callback

In this scenario, the replaying driver (JDBC or OCI) uses an application callback to set the initial state of the session during runtime and replay. The JDBC replay driver provides a connection initialization callback interface and methods to register and

unregister connection initialization callbacks in the `oracle.jdbc.replay.OracleDataSource` interface. For OCI and ODP.NET, register the TAF callback.

When registered, the initialization callback is executed every time a connection is borrowed from the pool and at each successful reconnection following a recoverable error. (This is true for the JDBC/UCP initialization callback and should be the same for TAF.) Using the same callback during both runtime and replay ensures that the same initialization is established at replay, as it was when the session was first established. An application is responsible for ensuring that the initialization actions are the same as that on the original connection before failover. If the callback invocation fails, then replay is disabled on that connection. Use the connection initialization callback only when the application has not implemented UCP and WebLogic Connection Labeling and the state cannot be restored automatically by setting either `FAILOVER_RESTORE=AUTO` for Transparent Application Continuity or `FAILOVER_RESTORE=LEVEL1` for manual Application Continuity.

Delaying the Reconnection in Application Continuity

By default, when Application Continuity initiates a failover, the driver attempts to recover the in-flight work at an instance where the service is available.

For recovering the work, the driver must establish a good connection with the instance. This reconnection can take some time if the database or the instance must be restarted before the service is relocated and published. So, the failover must be delayed until the service is available from another instance or database.

You must use the `FAILOVER_RETRIES` and `FAILOVER_DELAY` parameters to manage connecting and reconnecting. These parameters can work well in conjunction with a planned outage, for example, an outage that may make a service unavailable for several minutes. While setting the `FAILOVER_DELAY` and `FAILOVER_RETRIES` parameters, check the value of the `REPLAY_INITIATION_TIMEOUT` parameter first. The default value for this parameter is 900 seconds. A high value for the `FAILOVER_DELAY` parameter can cause replay to be canceled.

Parameter Name	Possible Value	Default Value
<code>FAILOVER_RETRIES</code>	Positive integer zero or above	30
<code>FAILOVER_DELAY</code>	Time in seconds	10

The following examples show various failover scenarios:

- [Creating Services on Oracle RAC with Application Continuity](#)
- [Modifying Services on Single-instance Databases to use Application Continuity](#)

Creating Services on Oracle RAC with Application Continuity

You can create services on Oracle RAC that utilize Transparent Application Continuity or manual Application Continuity.

You can create services that use Transparent Application Continuity, as follows:

For policy-managed databases:

```
$ srvctl add service -db codedb -service GOLD -serverpool ora.Srvpool -clbgoal SHORT
  -rlbgoal SERVICE_TIME -failover_restore AUTO -failoverretry 30 -failoverdelay 10
```

```
-commit_outcome TRUE -failovertype AUTO -replay_init_time 1800 -retention 86400
-notification TRUE
```

For administrator-managed databases:

```
$ srvctl add service -db codedb -service GOLD -preferred serv1 -available serv2 -
clbgoal SHORT
-rlbgoal SERVICE_TIME -failover_restore AUTO -failoverretry 30 -failoverdelay 10
-commit_outcome TRUE -failovertype AUTO -replay_init_time 1800 -retention 86400
-notification TRUE
```

You can create services that use manual Application Continuity, as follows::

For policy-managed databases:

```
$ srvctl add service -db codedb -service GOLD -serverpool ora.Srvpool -clbgoal SHORT
-rlbgoal SERVICE_TIME -failover_restore LEVEL1 -failoverretry 30 -failoverdelay 10
-commit_outcome TRUE -failovertype TRANSACTION -replay_init_time 1800 -retention
86400
-notification TRUE
```

For administrator-managed databases:

```
$ srvctl add service -db codedb -service GOLD -preferred serv1 -available serv2 -
clbgoal SHORT
-rlbgoal SERVICE_TIME -failover_restore LEVEL1 -failoverretry 30 -failoverdelay 10
-commit_outcome TRUE
-failovertype TRANSACTION -replay_init_time 1800 -retention 86400 -notification
TRUE
```

Modifying Services on Single-instance Databases to use Application Continuity

If you are using a single-instance database, then use the `DBMS_SERVICE` package to modify services.

For manual Application Continuity:

```
DECLARE
params dbms_service.svc_parameter_array;
BEGIN
params('FAILOVER_TYPE') := 'TRANSACTION';
params('REPLAY_INITIATION_TIMEOUT') := 1800;
params('RETENTION_TIMEOUT') := 86400;
params('FAILOVER_DELAY') := 10;
params('FAILOVER_RETRIES') := 30;
params('FAILOVER_RESTORE') := 'LEVEL1';
params('commit_outcome') := 'true';
params('aq_ha_notifications') := 'true';
dbms_service.modify_service('your service', params);
END;
/
```

For Transparent Application Continuity:

```
DECLARE
params dbms_service.svc_parameter_array;
BEGIN
params('FAILOVER_TYPE') := 'AUTO';
params('REPLAY_INITIATION_TIMEOUT') := 1800;
params('RETENTION_TIMEOUT') := 86400;
params('FAILOVER_DELAY') := 10;
params('FAILOVER_RETRIES') := 30;
```

```
params('FAILOVER_RESTORE'):= 'AUTO';
params('commit_outcome'):= 'true';
params('aq_ha_notifications'):= 'true';
dbms_service.modify_service('[your service]',params);
END;
/
```

Using Application Continuity with SQL*Plus

Use the `-ac` flag for SQL*Plus applications.

The `-ac` flag uses the standard `Dynamic` mode for session state, and disables `at COMMIT`.

- Use the `-ac` flag, as follows:

```
sqlplus -ac user/password@ACservice
```

Using Application Continuity for Planned Maintenance

For planned maintenance, the recommended approach is to drain requests from Oracle connection pools in combination with Application Continuity for those requests that do not complete. Instances do need to be stopped to switch over to the patched software.

This has the least impact when there is minimal recovery to complete.

To use Application Continuity for planned maintenance:

1. Use any FAN-aware pool, such as OCI, UCP, WebLogic Server, or ODP.NET Managed and Unmanaged Drivers.

The FAN planned event drains at request boundaries.

Note:

ODP.NET Managed Driver does not support Application Continuity.

2. Use the `srvctl relocate service` command to relocate the service from the instance without disrupting the sessions or, for a uniform service, use the `srvctl stop service` command on the instance (do not use the `-force` parameter).

The FAN planned event clears the idle sessions immediately and marks the active sessions to be released at check-in (end of request). This drains the sessions from the instance without disrupting work.

3. If not all sessions have checked in and the time to stop the instance has been reached, then stop the instance (abort).

For Application Continuity-enabled pools (UCP, WebLogic, Tuxedo, ODP.NET, and OCI), and any Java pool that adds `beginRequest/endRequest`, Application Continuity attempts to recover those remaining sessions.

4. Restart the instance and service.

Runtime load balancing, when enabled, balances the sessions back to the restored instance at the next request boundaries.

Running Without Application Continuity

Sometimes Application Continuity is not in effect because a disabling call has been issued.

Application Continuity is not in effect when it has not been started or when it has been disabled. If it has been disabled, it remains so through to the `endRequest` call.

Application Continuity is not started when the service property `FAILOVER_TYPE` does *not* have the value set to `TRANSACTION` or `AUTO`. For planned maintenance, set the `FAILOVER_TYPE` value to `TRANSACTION` or `AUTO`, beforehand; the setting applies to new connections, and existing connections retain their original service value.

Application Continuity is disabled for the current request when any of the following occurs:

- The application runs a statement that is restricted for Application Continuity (for example, `ALTER SYSTEM`).
- Application Continuity is explicitly disabled using `disableReplay`.
- A `COMMIT` statement is issued when the service parameter `session_state_consistency` is set to `Dynamic` (the default, when not using Transparent Application Continuity).
- An `endRequest` statement is issued until the next `beginRequest` is issued.
- The session is terminated or disconnected and the `NOREPLAY` keyword is specified.

Disabling Replay in Application Continuity

Replay occurs following a recoverable error but you can disable replay.

If an application has requests that the application does not want repeated, then the application can take a connection to a service that does not have Application Continuity enabled, or can explicitly call an API to disable replay for those requests. If you use Transparent Application Continuity, then side effects are detected and disabled automatically. You do not need to understand the application or to disable requests with side effects.

When using manual Application Continuity all calls are replayed. If an application uses `UTL_SMTP`, for example, and does not want messages repeated, then the application could use a connection to a different service, or use the `disableReplay` API on Java, or `OCIRequestDisableReplay` API for OCI. All other requests continue to be replayed.

For applications with external actions (for example, autonomous transactions or using `UTL_HTTP` to issue an SOA call), Application Continuity remains transparent if the application's correctness is preserved when these external actions are replayed after a failure.

The following rules are generic. They apply to all applications that replay work, including Application Continuity, and TAF (release 12.2 and after).

- [Application Calls Autonomous Transactions, External PL/SQL, or Java Actions that Should Not Be Repeated](#)
- [Application Synchronizes Independent Sessions](#)
- [Application Uses Time at the Middle Tier in the Execution Logic](#)

- [Application Assumes that ROWIDs Do Not Change](#)
- [Application Assumes that Location Values Do Not Change](#)

Application Calls Autonomous Transactions, External PL/SQL, or Java Actions that Should Not Be Repeated

Autonomous transactions, external PL/SQL calls, and Java callouts can have side effects that are separate from the main transaction, and these side effects are replayed unless you specify otherwise.

Examples of side effects separate from the main transaction include writing to an external table, sending email, forking sessions out of PL/SQL (including calls to `UTL_HTTP`, `UTL_URL`, `UTL_FILE`, `UTL_FILE_TRANSFER`, `UTL_SMPT`, `UTL_TCP`, `UTL_MAIL`, `DBMS_PIPE`, or `DBMS_ALERT`) or Java (including executing a shell script in the form `Process proc = rt.exec(command);`), transferring files, and accessing external URLs. Actions such as these leave persistent side effects. PL/SQL messaging and Java callouts can leave persistent results behind. For example, if a user walks away part way through some work without committing and the session times out or the user issues Ctrl+C, the foreground or a component fails; the main transaction rolls back while the side effects may have been applied. (For more information about side effects, see [Potential Side Effects of Application Continuity](#).)

Application developers decide whether to allow replay for external actions. Examples include using `UTL_HTTP` to issue an SOA call, or `UTL_SMTP` to send a message, or `UTL_URL` to access a website. If such external actions must not be replayed, use a connection without AC, or use one of the disable Replay APIs.

Application Synchronizes Independent Sessions

You must not configure an application for replay if the application synchronizes independent sessions using volatile entities that are held until `COMMIT`, `ROLLBACK`, or session loss. For example, the application might synchronize multiple sessions connected to several data sources that are otherwise inter-dependent using resources such as a database lock. This synchronization may be acceptable if the application is only serializing these sessions and understands that any session may fail. However, if the application assumes that a lock or any other volatile resource held by one data source implies exclusive access to data on the same or a separate data source from other connections, then this assumption may be invalidated when replaying.

During replay, the client driver is not aware that the sessions are dependent on one session holding a lock or other volatile resource. You can also use pipes, buffered queues, stored procedures taking a resource (such as a semaphore, device, or socket) to implement the synchronization that are lost by failures.

Application Uses Time at the Middle Tier in the Execution Logic

You must not configure an application for replay if the application uses the wall clock at the middle tier as part of the execution logic. The client driver does not repeat the middle-tier time logic, but uses the database calls that execute as part of this logic. For example, an application using middle-tier time might assume that a statement executed at Time T1 is not reexecuted at Time T2, unless the application explicitly does so.

Application Assumes that ROWIDs Do Not Change

If an application caches ROWIDs, then access to these ROWIDs might be invalidated due to database changes. Although a ROWID uniquely identifies a row in a table, a ROWID might change its value in the following situations:

- The underlying table is reorganized.
- An index is created on the table.
- The underlying table is partitioned.
- The underlying table is migrated.
- The underlying table is exported and imported using EXP/IMP/DUL.
- The underlying table is rebuilt using Golden Gate or Logical Standby or other replication technology.
- The database of the underlying table is flashed back or restored.

Oracle does not recommend, in general, that an application store ROWIDs for later use because the corresponding row might either not exist or contain completely different data. Note that ROWIDs do not prevent using Application Continuity. Replays can be rejected.

Application Assumes that Location Values Do Not Change

`SYSCONTEXT` options comprise a location-independent set such as National Language Support (NLS) settings, `ISDBA`, `CLIENT_IDENTIFIER`, `MODULE`, and `ACTION`, and a location-dependent set that uses physical locators. Typically, an application does not use the physical identifiers, except in testing environments. If physical locators are used in mainline code, then the replay finds the mismatch and rejects it. However, it is acceptable to use physical locators between requests (before `beginRequest`) or in callbacks. A common issue is for QA to modify test applications to select `V$INSTANCE`. As `V$INSTANCE` can change, only put this check in the callback or select the instance locally at the client and not from the database.

Example

```
select
  sys_context('USERENV','DB_NAME')
,sys_context('USERENV','HOST')
,sys_context('USERENV','INSTANCE')
,sys_context('USERENV','IP_ADDRESS')
,sys_context('USERENV','ISDBA')
,sys_context('USERENV','SESSIONID')
,sys_context('USERENV','TERMINAL')
,sys_context('USERENV','SID')
from dual;
```

Terminating or Disconnecting a Session Without Replay

If Application Continuity is configured and if a DBA terminates or disconnects a session by using the `ALTER SYSTEM KILL SESSION` or `ALTER SYSTEM DISCONNECT SESSION` statement, then Application Continuity, by default attempts, to recover the session. However, if you *do not* want the session to be replayed, then use the `NOREPLAY` keyword, as follows:

```
alter system kill session 'sid, serial#, @inst' noreplay;

alter system disconnect session 'sid, serial#, @inst' noreplay

$ srvctl stop service -db orcl -instance orcl2 -drain_timeout 60 -stopoption
immediate -force -noreplay

$ srvctl stop service -db orcl -node myode3 -noreplay -drain_timeout 60 -stopoption
immediate -force

$ srvctl stop instance -node mynode3 -drain_timeout 60 -stopoption immediate -force -
noreplay
```

To terminate all sessions running on the local instance (rather than only one session) and not have the sessions replayed, you can also use the `DBMS_SERVICE.DISCONNECT_SESSION` PL/SQL procedure and specify `NOREPLAY` for the `disconnect_option` parameter.

Mutable Functions and Application Continuity

When a request is replayed, the default and desired treatment of mutable objects can vary.

A **mutable function** is a nondeterministic function that can obtain a new value every time it is called. An example of a mutable function is a call to the `SYSTIMESTAMP` function. Client applications using Application Continuity can determine whether to keep the original value for mutable functions if the request is replayed.

Support for keeping mutable function values is currently provided for `SYSDATE`, `SYSTIMESTAMP`, `SYS_GUID`, and `sequence.NEXTVAL`. If the original values are not kept and if different values for these mutable objects are returned to the client, then replay is rejected because the client observes different results. If the application can use original values, then configure mutable functions using the `KEEP` clause for owned sequences and `GRANT KEEP` for other users. (Most applications need sequence values to be kept at replay, for bind variable consistency.)

Note:

Keeping `SYS_GUID` values is supported only for serial execution plans. When parallel query is used, Application Continuity is not able to restore original values for `SYS_GUID`.

Table 6-4 shows examples of the treatment of mutable functions by products during replay. (Actual implementation depends on specific products and releases.)

Table 6-4 Example Treatment of Mutable Objects by Products During Replay

Mutable Function	Product 1	Product 2	Product 3
<code>SYSDATE</code> , <code>SYSTIMESTAMP</code>	Original	Original	Current
Sequence <code>NEXTVAL</code> and <code>CURRVAL</code>	Original	Original	(Not applicable)

Table 6-4 (Cont.) Example Treatment of Mutable Objects by Products During Replay

Mutable Function	Product 1	Product 2	Product 3
SYS_GUID	Original	(Not applicable)	(Not applicable)

To allow Application Continuity to keep and use original function results at replay:

- The database user running the application might have the `KEEP DATE TIME` and `KEEP SYSGUID` privileges granted, and the `KEEP SEQUENCE` object privilege on each sequence whose value is to be kept. For example:

```
grant KEEP DATE TIME to user2;
grant KEEP SYSGUID to user2;
grant KEEP SEQUENCE on sales.seq1 to user2;
```

Notes:

- `GRANT ALL ON object` *does not* include (that is, does not grant the access provided by) the `KEEP DATE TIME` and `KEEP SYSGUID` privileges, and the `KEEP SEQUENCE` object privilege.
 - Grant privileges related to mutable function support only to application users, and to each application user grant only the necessary privileges.
 - Do *not* grant DBA privileges to database users running applications for which you want replay to be enabled.
- Sequences in the application can use the `KEEP` attribute, which keeps the original values of `sequence.NEXTVAL` for the sequence owner, so that the keys match during replay. Most applications need sequence values to be kept at replay. The following example sets the `KEEP` attribute for a sequence (in this case, one owned by the user executing the statement; for others, use `GRANT KEEP SEQUENCE`):

```
SQL> CREATE SEQUENCE my_seq KEEP;
SQL> -- Or, if the sequence already exists but without KEEP:
SQL> ALTER SEQUENCE my_seq KEEP;
```

Note:

Specifying `ALTER SEQUENCE ... KEEP/NOKEEP` applies to the owner of the sequence. It does not affect other users (not the owner) that have the `KEEP SEQUENCE` object privileges. If you want `NOKEEP` for all users, then be sure *not* to grant the `KEEP SEQUENCE` object privilege to these users (or to revoke it from each user if the privilege has been granted).

- To keep function results (for named functions) at replay, the DBA must grant `KEEP` privileges to the user invoking the function. This security restriction ensures that it is valid for replay to save and restore function results for code that is not owned by that user.

The following additional considerations apply to granting privileges on mutable functions:

- If a user has the `KEEP` privilege granted on mutable function values, then the objects inherit mutable access when the `SYS_GUID`, `SYSDATE`, and `SYSTIMESTAMP` functions are called.
- If the `KEEP` privilege is revoked on mutable values on a sequence object, then SQL or PL/SQL blocks using that object do not allow mutable collection or application for that sequence.
- If granted privileges are revoked between runtime and failover, then the mutable values that are collected are not applied for replay.
- If new privileges are granted between runtime and failover, then mutable values are not collected and these values are not applied for replay.

Administering Mutable Values

To keep function results at replay, you must grant `KEEP` privileges to the user invoking the function.

Grant and Revoke Interface

To grant and revoke permission to keep [mutables](#) for Oracle Dates and `SYS_GUID`:

```
GRANT [KEEP DATE_TIME | KEEP SYS_GUID]...[to USER]
REVOKE [KEEP DATE_TIME | KEEP SYS_GUID]...[from USER]
```

For example, for possible Oracle E-Business Suite usage with original dates:

```
GRANT KEEP DATE_TIME, KEEP SYS_GUID to [custom user];
GRANT KEEP DATE_TIME, KEEP SYS_GUID to [apps user];
```

Granting Permission to Keep Mutables for Oracle Sequences

To grant permission to the owner of the sequence:

```
CREATE SEQUENCE [sequence object] [KEEP|NOKEEP];
ALTER SEQUENCE [sequence object] [KEEP|NOKEEP];
```

The preceding commands keep the original values of `sequence.nextval` for replaying so that keys match.

To grant and revoke permission for others using the sequence:

```
GRANT KEEP SEQUENCE...[to USER] on [sequence object];
REVOKE KEEP SEQUENCE...[from USER] on [sequence object];
```

For example, for possible Oracle E-Business Suite usage with original sequence values:

```
GRANT KEEP SEQUENCE to [apps user] on [sequence object];
GRANT KEEP SEQUENCE to [custom user] on [sequence object];
```

Rules for Grants on Mutables

- If you grant all on an object for a user, then mutables are excluded. Mutables require explicit grants. Oracle does not support granting mutables to the users supplied or created by Oracle Database, such as `SYS`, `AUDSYS`, `GSMUSER`, and `SYSTEM`.

- The DBA role includes mutable permission.
- If a user has mutables granted, then the objects inherit mutable access when the mutable functions are called (in `SYS_GUID`, `SYSDATE` and `SYSTIMESTAMP`).
- If keeping mutables on a sequence object is revoked, then SQL or PL/SQL commands using that object does not allow mutable collection or application for that sequence.
- If grants are revoked between run time and failover, then the mutables that were collected are not applied.
- If grants are granted between run time and failover, then mutables are not collected and so none are applied.

Protection-Level Statistics

Use the statistics for request boundaries and protection level to monitor the level of coverage.

Application Continuity collects statistics from the system, the session, and the service, enabling you to monitor your protection levels. The statistics are available in `V$SESSTAT`, `V$SYSSTAT`, and, when service statistics are enabled, in `V$SERVICE_STATS`. These statistics are saved in the Automatic Workload Repository and are available in Automatic Workload Repository reports. Statistics include:

- Requests completed per second
- User calls in a request
- Protected user calls

The output is similar to the following:

Statistic	Total	per Second	per Trans
cumulative requests	177,406	49.2	5.0
cumulative user calls in request	493,329	136.8	13.8
cumulative user calls protected	493,329	136.8	13.8

To enable protection-level statistics, use (`_request_boundaries = 3`).

Session State Consistency

Session state consistency describes how non-transactional state is changed during a request.

Oracle recommends that you set `session_state_consistency` to `AUTO` available with Transparent Application Continuity, which tracks and manages session states. If you choose to use Transparent Application Continuity, then you do not have to do anything else to ensure session state consistency.

You can set `session_state_consistency` to `DYNAMIC` or `STATIC` for manual Application Continuity. Set `session_state_consistency` to `DYNAMIC` or `STATIC` if you fully understand the application, and the application is not expected to change from the value set.

Examples of session state are NLS settings, optimizer preferences, event settings, PL/SQL global variables, temporary tables, advanced queues, LOBs, and result cache. If non-transactional values change in committed transactions, then use the

default value, `DYNAMIC` (`session_state_consistency` is a service level attribute, the default value of which is `DYNAMIC`).

Using `DYNAMIC` mode, after a `COMMIT` has executed, if the state was changed in that transaction, then it is not possible to replay the transaction to reestablish that state if the session is lost. Applications can be categorized depending on whether the session state after the initial setup is static or dynamic, and hence whether it is correct to continue past a `COMMIT` operation.

`DYNAMIC` mode is appropriate for almost all applications. If you are unsure, then use `DYNAMIC` mode. If your customers or users can modify your application, then you must use `DYNAMIC` mode.

 **Note:**

Set `session_state_consistency` to `AUTO` or `STATIC` for long-running, stateless applications. Do not set `session_state_consistency` to `STATIC` for applications that are not stateless. Unless you require manual Application Continuity, Oracle recommends setting `session_state_consistency` to `AUTO`.

This section includes the following topics:

- [Auto Session State Consistency](#)
- [Dynamic Session State Consistency](#)
- [Static Session State Consistency](#)

Auto Session State Consistency

When you set `session_state_consistency` to `AUTO`, Transparent Application Continuity tracks and records session and transactional states so the database session can be recovered following recoverable outages.

Beginning with Oracle Database 18c, Oracle recommends that most applications set `session_state_consistency` to `AUTO`. When set to `AUTO`, a state-tracking infrastructure categorizes session state usage as the application issues user calls. This ensures consistency for the application now and for future changes. Setting `session_state_consistency` to `AUTO` is the only value permitted for Transparent Application Continuity.

Dynamic Session State Consistency

A session has **dynamic** state if the session state values are not fully restored by `FAILOVER_RESTORE`, or by adding the initialization callback.

Once the first transaction completes, failover is internally disabled until the next request starts. In `Dynamic` session state consistency mode, state changes occur during the request and replay is enabled at the beginning of the next request.

Set the session state consistency mode to `Dynamic` if the nontransactional session state changes while transactions are executing. Examples of nontransactional session state that can change at runtime are `ALTER SESSION`, PL/SQL global variables, `SYS_CONTEXT`, and temporary table contents. If the application changes nontransactional state inside transactions and commits, this state cannot be replayed and the state setting must be

`Dynamic`. When using `Dynamic` mode for Application Continuity, replay is disabled at `COMMIT` until the next request begins. `Dynamic` is the default value.

The nontransactional session state (NTSS) changes during a request when the session state consistency mode is `Dynamic`.

Replay (that is, Application Continuity) is enabled at the `beginRequest` call, and is disabled on a `COMMIT`, an `endRequest` call, or a restricted call. Following is the step logic for three application scenarios:

- No transaction
- A transaction with `COMMIT` as the last statement
- A transaction with an embedded `COMMIT` statement

For the request with **no transaction**, the logical steps are as follows:

1. Check out.
2. Begin request and enable replay.
3. Issue one or more `SELECT` statements and perhaps other PL/SQL statements.
4. Other actions.
5. Check in.
6. End request and disable replay.

For the request with **a transaction with `COMMIT` as the last statement**, the logical steps are as follows:

1. Check out.
2. Begin request and enable replay.
3. Issue one or more `SELECT` statements and perhaps other PL/SQL statements.
4. The transaction begins.
5. Other actions.
6. Commit (which disables replay).
7. Check in.
8. End request.

For the request with **a transaction with an embedded `COMMIT` statement**, the logical steps are as follows:

1. Check out.
2. Begin request and enable replay.
3. Issue one or more `SELECT` statements and perhaps other PL/SQL statements.
4. The transaction begins.
5. Other actions.
6. Commit (which disables replay).
7. Other actions, during which Application Continuity is not covering the application.
8. Check in.
9. End request.

Static Session State Consistency

`Static` mode is used for long running stateless applications. Do not use `Static` mode for applications that are not stateless.

Set the session state consistency mode to `Static`, only if all non-transactional state changes, such as NLS settings, `SYS_CONTEXT`, PL/SQL variables, and optimizer preferences, are set as part of the initialization once per request, and if this session state does not change during transactions. The settings can be established once per connection at connection establishment when using `FAILOVER_RESTORE=LEVEL1`, a callback, or labels, for example, or at each checkout from a pool.

When using `Static` mode for Application Continuity, transactional failover continues beyond the first transaction of a request. This is useful for applications that set `beginRequest` once and run long processing operations such as batch jobs, and long reports.

`Static` mode is not supported for applications that use calls that change non-transactional state in transactions. Specific examples of such calls include:

- PL/SQL subprograms
- `SYS_CONTEXT`
- `ALTER SESSION`

Specify `static` mode with caution. Use `static` mode only when the application does not change the non-transactional session state inside transactions. Declaring the session state consistency mode as `Static` indicates that it is safe to continue beyond the first `COMMIT` in a request. Dynamic mode is appropriate for most applications. Do *not* use `static` mode if users or customers can modify or customize the application.

The non-transactional session state remaining constant (that is, not changing) during a request when the session state consistency mode is `Static`.

Replay (that is, Application Continuity) is enabled at the `beginRequest` call, and is disabled on a restricted call, on a `disableReplay` or `OCIRequestDisableReplay` call, or on an `endRequest` call.

Following is the step logic for three application scenarios:

- No transaction
- One or more transactions each ending with `COMMIT` as the last statement
- A transaction with a `COMMIT` statement followed by a transaction with a restricted call that disables Application Continuity

For the request with **no transaction**, the logical steps are as follows:

1. Check out.
2. Begin request and enable replay.
3. Issue one or more `SELECT` statements and perhaps other PL/SQL statements.
4. Other actions.
5. Check in.
6. End request and disable replay.

Replay is disabled at `endRequest`, at a restricted call, and for an explicit `disableReplay` or `OCIRequestDisableReplay` call.

For the request with **one or more transactions (each with COMMIT as the last statement)**, the logical steps are as follows:

1. Check out.
2. Begin request and enable replay.
3. Issue one or more `SELECT` statements and perhaps other PL/SQL statements.
4. The transaction begins.
5. The transaction commits.
6. The transaction is purged.
(For each additional transaction, steps 4 through 6 occur.)
7. Other actions.
8. Check in.
9. End request.

Replay is disabled at `endRequest`, at a restricted call, and for an explicit `disableReplay` or `OCIRequestDisableReplay` call.

For the request with **a transaction with a COMMIT followed by a transaction with a restricted call**, the logical steps are as follows:

1. Check out.
2. Begin request and enable replay.
3. Issue one or more `SELECT` statements and perhaps other PL/SQL statements.
4. The transaction begins.
5. The transaction commits.
6. The transaction is purged.
7. The second transaction begins.
8. The transaction makes a restricted call, which causes Application Continuity to be disabled.
9. The transaction is purged.
10. Other actions
11. Check in.
12. End request.

Replay is disabled at `endRequest`, at a restricted call, and for an explicit `disableReplay` or `OCIRequestDisableReplay` call.

Potential Side Effects of Application Continuity

When you use Application Continuity with `FAILOVER_TYPE` set to `TRANSACTION`, statements that leave side effects are replayed.

 **Note:**

As an application owner you can elect to disable replay for requests that contain side effects that you do not want to repeat. The simplest way to disable side effects is to use Transparent Application Continuity (set `FAILOVER_TYPE` to `AUTO`, which disables side effects for you).

Application Continuity replays PL/SQL chronologically to restore database state. This serves to rebuild the session as if the user submission was delayed. Most applications want the full state rebuilt as if the submission was repeated, such as writing a report or completing some auditing. However, the actions that are replayed to build the state might include some for which you want to take action to accommodate or mitigate the effects of the replay. Some applications elect to disable replay for requests that contain calls that they do not want to repeat.

Examples of actions that create side effects include the following:

- `DBMS_ALERT` calls (email or other notifications)
- `DBMS_FILE_TRANSFER` calls (copying files)
- `DBMS_PIPE` and `RPC` calls (to external sources)
- `UTL_FILE` calls (writing text files)
- `UTL_HTTP` calls (making HTTP callouts)
- `UTL_MAIL` calls (sending email)
- `UTL_SMTP` calls (sending SMTP messages)
- `UTL_TCP` calls (sending TCP messages)
- `UTL_URL` calls (accessing URLs)

For applications with external actions (such as autonomous transactions or using `UTL_HTTP` to issue a service-oriented application (SOA) call), Application Continuity is transparent when the application is satisfied with replaying external actions, such as resending email, auditing, and transferring a file.

Restrictions and Other Considerations for Application Continuity

Be aware of these restrictions and considerations when using Application Continuity.

- Application Continuity excludes :
 - JDBC OCI driver (type 2)
 - ODP.NET, Managed Driver
 - OLE DB
 - ODBC
- For applications using JDBC, there is no support for `oracle.sql` deprecated concrete classes: `BLOB`, `CLOB`, `BFILE`, `OPAQUE`, `ARRAY`, `STRUCT`, or `ORADATA`.
- For streams arguments, replay is on a "best effort" basis. For example, if the application is using physical addresses, the address has gone with the outage and

cannot be repositioned. JDBC stream setters (such as `setBinaryStream`), for example, cause replay to be disabled.

- The replay target database must be in the same database cluster (Oracle RAC, Oracle Data Guard, Oracle Active Data Guard, or Oracle Multitenant) as the source database.

To protect the integrity of business transactions, Application Continuity does not replay if the target is a different database or if it is the same database or same pluggable database but with data loss, such as one flashed back, recovered incompletely by media recovery, or opened by Oracle Data Guard earlier in time.

- For OCI and ODP.NET, in Oracle Database 12c release 2 (12.2.0.1), Application Continuity on OCI driver excludes ADTs, advance queues, and some LOB APIs. These exclusions do not apply to Java.
- If a statement cache at the application server level is enabled (for example, the WebLogic or third-party application server statement cache), this must be disabled when the replay is used. Instead, configure the JDBC statement cache, which performs better because it is optimized for JDBC and Oracle and because it supports Application Continuity. Use `oracle.jdbc.implicitstatementcachesize=nnn`.
- Starting with Oracle Database 12 release 2 (12.2), replay is supported for the XA data source for Java and ODP.NET, Unmanaged Driver. Replay supports local transactions. Replay silently disables when two-phase is used. This enables Application Continuity to work very well with promotable XA and with applications using the XA data source and mostly not using XA.
- Replay is disabled if a request issues an `ALTER SYSTEM` or `ALTER DATABASE` statement.
- Replay is disabled at a request level for `ALTER SESSION` statements that are deemed unsafe to rebuild the session. These include statements for setting support-level events, and disabling and enabling `COMMIT IN PROCEDURE` and `GUARD`.

However, `ALTER SESSION` statements at an application level are supported for replay. These include statements for globalization support (NLS) settings, stored and private outlines, setting the container (CDB/PDB), SQL trace, and PL/SQL warnings.

- Replay is not supported if you are using Oracle Active Data Guard with read/write database links back to the primary database. This is a security restriction from Transaction Guard.
- Replay does not apply for failure of a parallel query call when this is a statement-level failure. For example, replay would not occur after an `ORA-12805:parallel query server died unexpectedly` error for a call failure encountered during an instance or node failure or memory issue.
- For Java only, replay does not support DRCP. Dedicated and Shared Servers are supported.
- The request uses 2PC XA. Starting with Oracle Database 12c release 2 (12.2), Application Continuity is supported for promotable XA and using XA data sources, while XA is not in use.

 **Note:**

If you are creating clones of databases by splitting disk images (for example, BCVs) or by cloning so it is a "different" database to make a logical standby or logical copy that is not a physical or Oracle Active Data Guard database, then `nid` **must** be used to change the DBID to differentiate the databases. For information about `nid` program usage, see these My Oracle Support notes: *How to Change the DBID and the DBNAME by using NID* (Doc Id 224266.1) and *Changing DBNAME and DBID of Oracle RAC Database Using NID* (Doc Id 464922.1).

Transaction Guard for Improving Client Failover

Transaction Guard prevents a transaction being replayed by Application Continuity from being applied more than once.

Failing to recognize that the last submission has committed, or that it shall commit sometime soon, or that the last submission has not run to completion, is a problem for applications. It can cause users who resubmit or applications that use their own replay to issue duplicate requests, repeating changes that are already committed to the database, and other forms of logical corruption. Transaction Guard can be used to solve this problem.

Application Continuity automatically enables and uses Transaction Guard, but you can also enable Transaction Guard independently. If the application has implemented an application-level replay, then it requires the application to be integrated with Transaction Guard to provide idempotence.

In Oracle Database 12c, Transaction Guard provides a new, fully integrated tool for applications to use to achieve idempotence automatically and transparently, and in a manner that scales. Transaction Guard uses Logical Transaction ID (LTXID) to avoid submitting duplicate transactions. This is referred to as *transaction idempotence*. The LTXID persists on commit and is reused following a rollback. During normal runtime, a LTXID is automatically held in the session at both the client and server for each database transaction. At commit, the LTXID is persisted as part of committing the transaction and the next LTXID to use is returned to the client.

Transaction Guard for XA Transactions

Transaction Guard also supports XA-based transactions, which are transactions that are an option for transaction managers, such as Oracle WebLogic Server, Oracle Tuxedo, and MicroSoft Transaction Server (exposed to Oracle Database through Oracle ODP.NET).

Transaction Guard support for XA transactions provides safe replay following recoverable outages for XA transactions on Oracle WebLogic Server. With the addition of XA support, Oracle WebLogic Server can provide replay with idempotence enforced using Transaction Guard.

This section includes the following topics:

- [Transaction Guard Configuration Checklist](#)
- [Configuring Services for Transaction Guard](#)

- [Failing Over OCI Clients with TAF](#)

Transaction Guard Configuration Checklist

Before configuring services for Transaction Guard, use the following configuration checklist:

- Grant permission to the application user who will call `GET_LTXID_OUTCOME`, as follows:


```
GRANT EXECUTE ON DBMS_APP_CONT to user_name;
```
- Locate and define the transaction history table for optimal performance.

The transaction history table (`LTXID_HIST`) is created, by default, in the `SYSAUX` tablespace when you create or upgrade an Oracle Database. New partitions are added when you add instances, using the storage of the last partition. If the location of transaction history table is not optimal for performance, then you can move it to another tablespace and create partitions there. For example, the following statement moves the transaction history table to a tablespace named `FastPace`:

```
ALTER TABLE LTXID_TRANS move partition LTXID_TRANS_1 tablespace FastPace
  storage ( initial 10G next 10G minextents 1 maxextents 121 );
```
- Set values for the `-commit_outcome` and `-retention` service parameters.
- If you are using Oracle RAC, Oracle Data Guard, or Oracle Active Data Guard, then Oracle recommends that you use `FAN` for fast notification of an outage.

Configuring Services for Transaction Guard

To configure services to use Transaction Guard, set the following service parameters:

- **`-commit_outcome`:** Set the `-commit_outcome` service parameter to `TRUE`. This service parameter determines whether the transaction [commit outcome](#) is accessible after the `COMMIT` has executed and an outage has occurred. While Oracle Database has always made `COMMIT` durable, Transaction Guard makes *the outcome* of the `COMMIT` durable, and is used by applications to enforce the status of the last transaction executed before an outage.
- **`-retention`:** Use the `-retention` service parameter with `-commit_outcome`. This service parameter determines the amount of time, in seconds, that the `COMMIT` outcome is retained. Oracle recommends that most installations use the default value.

The following `SRVCTL` command configures a policy-managed service named `sales` for Transaction Guard:

```
$ srvctl add service -db crm -service sales -serverpool spool_1
  -commit_outcome TRUE -retention 86400 -notification TRUE
```

The following `SRVCTL` command configures an administrator-managed service named `sales` for Transaction Guard:

```
$ srvctl add service -db crm -service sales -preferred crm_1,crm_2
  -available crm_3,crm_4 -commit_outcome TRUE -retention 86400
  -notification TRUE
```

You can also modify an existing service to configure it for Transaction Guard by using the `srvctl modify service` command.

 **Note:**

Do not use the default database service, the service which has the name set to the value of `db_name` or `db_unique_name`. The default service is used for administrative purposes and does not have the same properties as user-created services.

Failing Over OCI Clients with TAF

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 transparent application failover (TAF) for the connection, then Oracle Database replays the session at 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 also 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.

You can define a TAF policy for all users of a service by defining the `-failovermethod` and `-failovertime` parameters. You can further define the TAF policy by setting the number of times that a failed session attempts to reconnect to the service and how long it should wait between reconnection attempts using the `-failoverretry` and `-failoverdelay` parameters, respectively.

To define a TAF policy for a service, use `SRVCTL` as in the following example, where the service name is `tafconn.example.com` and the database name is `CRM`:

```
$ srvctl modify service -db crm -service tafconn.example.com -failovermethod BASIC  
-failovertime SELECT -failoverretry 10 -failoverdelay 30
```

OCI applications with TAF enabled should use FAN high availability events for fast connection failover.

TAF Supports Transaction Guard and `FAILOVER_RESTORE`

When you are using Transaction Guard, TAF manages the errors for the developers. When you use both TAF and Transaction Guard, developers can use the TAF errors to roll back and safely resubmit or return uncommitted transactions (for TAF error codes `ORA-25402`, `ORA-25408`, `ORA-25405`).

When you are using `FAILOVER_RESTORE`, TAF automatically restores common states, which avoids the need for a callback for most applications.

7

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 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 RAC](#)
- [Archiving Mode in Oracle RAC](#)
- [Configuring the RMAN Snapshot Control File Location](#)
- [Configuring RMAN to Automatically Backup the Control File and SPFILE](#)
- [Crosschecking on Multiple Oracle RAC Nodes](#)
- [Configuring Channels for RMAN in Oracle RAC](#)
- [Managing Archived Redo Logs Using RMAN in Oracle RAC](#)
- [Archived Redo Log File Conventions in Oracle RAC](#)
- [RMAN Archiving Configuration Scenarios](#)
- [Monitoring the Archiver Processes](#)

Overview of Configuring RMAN for Oracle RAC

RMAN enables you to back up, restore, and recover data files, control files, server parameter files (SPFILEs) and archived redo log files. RMAN is included with 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.

Archiving Mode in Oracle RAC

In order for redo log files to be archived, the Oracle RAC database must be in `ARCHIVELOG` mode.

You can run the `ALTER DATABASE` SQL statement to change the archiving mode in Oracle RAC, because 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:**

- The `ARCHIVELOG` mode is set at the database level, not the instance level. Either all instances archive or none do.
- You can also change the archive log mode by using the Recovery Settings page in the **Maintenance** tab of the Oracle Enterprise Manager Oracle RAC Database Home Page.

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 RMAN.

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.

You can specify a [cluster file system](#) or a raw device destination for the location of your snapshot control file. This file is shared across all nodes in the [cluster](#) and *must* be accessible by all nodes in the cluster. 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 sets the configuration for the location of the snapshot control file for every instance of your [cluster database](#). Therefore, ensure that the `ORACLE_HOME/dbs/scf` directory is shared by 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.

To delete a snapshot control file you must first change the snapshot control file location, then delete the file at the older location, as follows:

```
CONFIGURE SNAPSHOT CONTROLFILE NAME TO 'new_name';  
DELETE COPY OF CONTROLFILE;
```

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, because the default location for the SPFILE must be available to all nodes in your Oracle RAC database.

 **Note:**

If you back up the control file using the SQL*Plus `ALTER DATABASE` command, then you must also create the control file backup on a device shared by all nodes.

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 Oracle Enterprise Manager to use the RMAN features.

Crosschecking on Multiple Oracle RAC 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 to 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.

Configuring Channels for RMAN in Oracle RAC

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 Node](#)

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 Node

To configure one RMAN channel for each policy-managed Oracle RAC database instance, use the following syntax:

```
CONFIGURE CHANNEL DEVICE TYPE sbt CONNECT '@racinst_1'  
CONFIGURE CHANNEL DEVICE TYPE sbt CONNECT '@racinst_2'  
...
```

After this one-time configuration step, you can issue the `BACKUP` or `RESTORE` commands.

Managing Archived Redo Logs Using RMAN in Oracle RAC

When a node generates an archived redo log, Oracle Database always records the file name 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 file name on its file system, the file must be readable by any node that must access this archived redo log. For example, if `node1` archives a log to `/oracle/arc_dest/log_1_100_23452345.arc`, then `node2` can 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 must 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, only 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 the number of parallel threads.

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. During recovery, if the

archived log destinations are visible from the node that performs the recovery, then Oracle Database can successfully recover the archived log data.

Archived Redo Log File Conventions in Oracle RAC

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 file name extension.

Table 7-1 Archived Redo Log File Name Format Parameters

Parameter	Description	Example
<code>%r</code>	Resetlogs identifier, not padded	log_1_62_23452345
<code>%R</code>	Resetlogs identifier, left-zero-padded	log_1_62_0023452345
<code>%s</code>	Log sequence number, not padded	log_251
<code>%S</code>	Log sequence number, left-zero-padded	log_0000000251
<code>%t</code>	Thread number, not padded	log_1
<code>%T</code>	Thread number, left-zero-padded	log_0001

All of the file name format parameters for the archive redo logs, in either upper or lowercase, are mandatory for Oracle RAC. These parameters enable 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
...
```

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 redo log files.

This section includes the following topics:

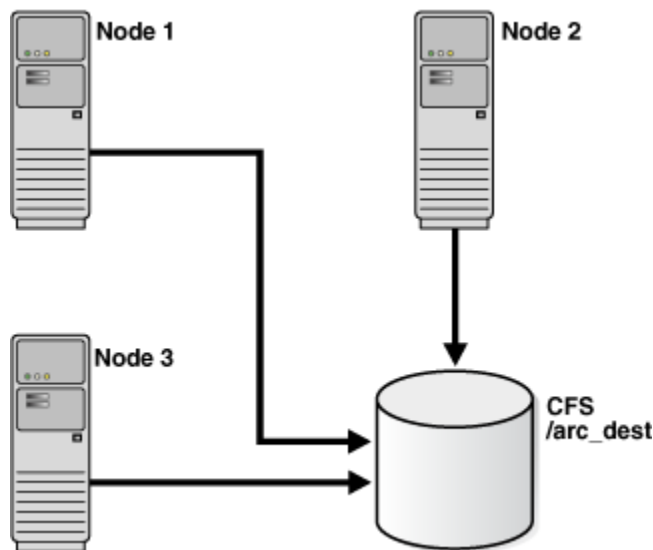
- [Oracle Automatic Storage Management and Cluster File System Archiving Scheme](#)
- [Noncluster File System Local Archiving Scheme](#)

Oracle Automatic Storage Management and Cluster File System Archiving Scheme

The preferred configuration for Oracle RAC is to use Oracle Automatic Storage Management (Oracle ASM) for a recovery area using a disk group for your recovery set that is different from the disk group used for your data files.

When you use Oracle ASM, it uses an Oracle Managed Files naming format. Alternatively, you can use a cluster file system archiving scheme. If you use a cluster file system, then each node writes to a single location on the cluster file system when archiving the redo log files. Each node can read the archived redo log files of the other nodes. For example, as shown in [Figure 7-1](#), if Node 1 archives a redo log file 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.

Figure 7-1 Cluster File System Archiving Scheme



Note:

The archive log naming format in this example is only for a cluster file system example.

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.

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 file name 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 (`/arc_dest`, in the following example). To configure this directory, set values for the `LOG_ARCH_DEST_1` parameter, as shown in the following example:

```
*.LOG_ARCHIVE_DEST_1="LOCATION=/arc_dest"
```

The following list shows archived redo log entry examples that would appear in the RMAN catalog or 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 node3  
/arc_dest/log_3_1563_23452345.arc <- thread 3 archived in node2  
/arc_dest/log_2_753_23452345.arc <- thread 2 archived in node1  
/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 the `/arc_dest` directory in the cluster file system, each node can read the logs written by itself and any other node.

Noncluster File System Local Archiving Scheme

When archiving locally to a noncluster file system, 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 archive directories on the other nodes.

In addition, if you are performing recovery and 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, to distribute the backup processing onto multiple nodes, the easiest method is to configure channels as described in the backup scenarios in "Managing Backup and Recovery".

 **Note:**

Because different file systems are used in a noncluster case, the archive log directories must be unique on each node. For example, `/arc_dest_1` is only available on `node1`, `/arc_dest_2` is only directly mounted on `node2`, and so on.

Then `node1` mounts `/arc_dest_2` from `node2` and `/arc_dest_3` from `node3` through NFS.

Initialization Parameter Settings for Noncluster File System Local Archiving

You can set the archiving destination values as follows in the initialization parameter file for either policy-managed or administrator-managed databases.

Set the `SID.LOG_ARCH_DEST` parameter for each instance using the SID designator, as shown in the following example:

```
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"
```

For policy-managed databases, manually create a node and instance binding to ensure that `sid1` always runs on the same node, as follows:

```
$ srvctl modify database -d mydb -n node1 -i sid1
$ srvctl modify database -d mydb -n node2 -i sid2
$ srvctl modify database -d mydb -n node3 -i sid3
```

The following list shows the possible archived redo log entries in the database control file. Note that any node can read archived redo logs from any of the threads, which must happen in order for the database to recover after a failure.

```
/arc_dest_1/log_1_1000_23435343.arc
/arc_dest_2/log_1_1001_23452345.arc <- thread 1 archived in node2
/arc_dest_2/log_3_1563_23452345.arc <- thread 3 archived in node2
/arc_dest_1/log_2_753_23452345.arc <- thread 2 archived in node1
/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 7-2](#), each of three nodes 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.

 **Note:**

The archive log destinations, similar to those shown in [Table 7-2](#), must be different on each node so that if you mount the NFS directory on a different node, then it does not conflict with an existing archive log directory

Table 7-2 UNIX/NFS Location Log Examples, Noncluster File System Local Archiving

Node	Reads 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 7-3](#).

Table 7-3 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.

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.

 **Note:**

If you use the `kill` command to stop the archiver process, then the database instance will fail.

8

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 Fast Recovery Area in Oracle RAC. The topics in this chapter include:

- [RMAN Backup Scenario for Noncluster File System Backups](#)
- [RMAN Restore Scenarios for Oracle RAC](#)
- [Instance Recovery in Oracle RAC](#)
- [Media Recovery in Oracle RAC](#)
- [Parallel Recovery in Oracle RAC](#)
- [Using a Fast Recovery Area in Oracle RAC](#)

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 data files. In Oracle RAC, data files are accessible from every node in the [cluster](#), so any node can restore archived redo log files.

RMAN Backup Scenario for Noncluster File System Backups

In a [noncluster file system](#) environment, each node can back up only to a locally mounted noncluster file system directory. For example, `node1` cannot access the archived redo log files on `node2` or `node3` unless you configure the network file system for remote access. If you configure a network file system file for backups, then each node backs up its archived redo logs to a local directory.

RMAN Restore Scenarios for Oracle RAC

This section describes the following common RMAN restore scenarios:

- [Restoring Backups from a Cluster File System](#)
- [Restoring Backups from a Noncluster File System](#)
- [Using RMAN or Oracle 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 noncluster scenarios.

Restoring Backups from a Cluster File System

The scheme that this section describes assumes that you are using the "Oracle Automatic Storage Management and Cluster File System Archiving Scheme". In this scheme, assume that `node3` performed the backups to a cluster file system. If `node3` 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 `node3` performed the backups but is unavailable, then configure a media management device for one of the remaining nodes and make the backup media from `node3` available to this node.

 **Note:**

If you configured RMAN as described in "Configuring Channels to Use Automatic Load Balancing", then, to load balance the channels across nodes, note that channels cannot be load balanced before at least one instance has successfully opened the database. This means that the channels will not be load balanced across the nodes during a full database restore. To achieve load balancing of channels for `RESTORE` and `RECOVER` commands, you can temporarily reallocate channels by running commands similar to the following:

```
run {  
  ALLOCATE CHANNEL DEVICE TYPE sbt C1 CONNECT '@racinst_1'  
  ALLOCATE CHANNEL DEVICE TYPE sbt C2 CONNECT '@racinst_2'  
  ...  
}
```

Restoring Backups from a Noncluster File System

The scheme that this section describes assumes that you are using the "Noncluster File System Local Archiving Scheme". In this scheme, each node archives locally to a different directory. For example, `node1` archives to `/arc_dest_1`, `node2` archives to `/arc_dest_2`, and `node3` 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 redo log files on 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 Oracle 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 Oracle Enterprise Manager to restore the 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 RAC

Instance failure occurs when software or hardware problems disable an instance. After instance failure, Oracle Database automatically uses the online redo logs to perform recovery as described in this section.

This section includes the following topics:

- [Single Node Failure in Oracle RAC](#)
- [Multiple-Node Failures in Oracle RAC](#)
- [Using RMAN to Create Backups in Oracle RAC](#)
- [Node Affinity Awareness of Fast Connections](#)
- [Deleting Archived Redo Logs after a Successful Backup](#)
- [Autolocation for Backup and Restore Commands](#)

Single Node Failure in Oracle RAC

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.

Applications that were running on a node before it failed continue running by using failure recognition and recovery. This provides consistent and uninterrupted service if hardware or software fails. 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 RAC

When multiple node failures occur, if one instance survives, then Oracle RAC performs instance recovery for any other instances that fail. If all instances of an Oracle RAC database fail, then Oracle Database 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 Database running in shared mode as it is for Oracle Database running in exclusive mode, except that one instance performs instance recovery for all of the failed instances.

Using RMAN to Create Backups in Oracle RAC

Oracle Database provides RMAN for backing up and restoring the database.

RMAN enables you to back up, restore, and recover data files, control files, SPFILES, and archived redo logs. RMAN is included with the Oracle Database 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 Oracle Automatic Storage Management (Oracle ASM). The procedures for using RMAN in Oracle RAC environments do not differ substantially from those for Oracle noncluster environments.

Channel Connections to Cluster Instances with RMAN

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 `dbauser/pwd@service_name`. If you configure the SQL Net service name with load balancing turned on, then the 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 'dbauser/pwd@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 'dbauser/pwd@mydb_1';  
CONFIGURE CHANNEL 2.. CONNECT 'dbauser/pwd@mydb_2';  
CONFIGURE CHANNEL 3.. CONNECT 'dbauser/pwd@mydb_3';
```

In the previous example, it is assumed that `mydb_1`, `mydb_2` and `mydb_3` 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, control file, data files and archived redo logs:

```
RUN
{
  ALLOCATE CHANNEL CH1 CONNECT 'dbauser/pwd@mydb_1';
  ALLOCATE CHANNEL CH2 CONNECT 'dbauser/pwd@mydb_2';
  ALLOCATE CHANNEL CH3 CONNECT 'dbauser/pwd@mydb_3';
  BACKUP DATABASE PLUS ARCHIVED LOG;
}
```

During a backup operation, if at least one channel allocated has access to the archived log, then RMAN automatically schedules the backup of the specific log on that channel. Because the control file, SPFILE, and data files 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 cluster file system archiving scheme, if every node writes the archived logs in the same cluster file system, then 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 instance on `node1` has the database mounted while the instances on `node2` and `node3` 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 data files than to other data files. RMAN automatically detects this situation, which is known as node affinity awareness. When deciding which channel to use to back up a particular data file, RMAN gives preference to the nodes with faster access to the data files that you want to back up. For example, if you have a three-node cluster, and if `node1` has faster read/write access to data files 7, 8, and 9 than the other nodes, then `node1` has greater node affinity to those files than `node2` and `node3`.

Deleting Archived Redo Logs after a Successful Backup

If you have configured the automatic channels as defined in section "Channel Connections to Cluster Instances with RMAN", then 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, if at least one channel allocated has access to the archived log, then RMAN automatically schedules 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 cluster file system archiving scheme, if every node writes to the archived logs on the same cluster file system, then 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 must 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 `node1`, while log 1002 is backed up to the drive attached to `node2`. If you then allocate channels that connect to each node, then the channel connected to `node1` can restore log 1001 (but not 1002), and the channel connected to `node2` can restore log 1002 (but not 1001).

Media Recovery in Oracle RAC

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 data files 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 noncluster environments.

The node that performs the recovery must be able to restore all of the required data files. 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 RAC

Oracle Database automatically selects the optimum degree of parallelism for instance, crash, and media recovery.

Oracle Database 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](#)

Parallel Recovery with RMAN

With RMAN's `RESTORE` and `RECOVER` commands, Oracle Database automatically makes parallel the following three stages of recovery:

Restoring Data Files

When restoring data files, 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 data files.

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 Database automatically selects the optimum degree of parallelism based on available CPU resources.

Disabling Parallel Recovery

You can override parallel recovery 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 in the database initialization parameter file, `SPFILE`, to 0 or 1.

Disabling Media Recovery Parallelism

Use the `NOPARALLEL` clause of the `RMAN RECOVER` command or the `ALTER DATABASE RECOVER` statement to force Oracle Database to use non-parallel media recovery.

Using a Fast Recovery Area in Oracle RAC

To use a fast recovery area in Oracle RAC, you must place it on an Oracle ASM disk group, on 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 fast 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.

Oracle Enterprise Manager enables you to set up a fast 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 Fast Recovery Area section of the page.
4. Click **Help** on this page for more information.

9

Cloning Oracle RAC to Nodes in a New Cluster

This chapter describes how to clone Oracle Real Application Clusters (Oracle RAC) database homes on Linux and UNIX systems to nodes in a new cluster.

This chapter describes a noninteractive cloning technique that you implement with 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, 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 Oracle 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 Enterprise Manager Cloud 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 includes the following topics:

- [Introduction to Cloning Oracle RAC](#)
- [Preparing to Clone Oracle RAC](#)
- [Deploying Oracle RAC Clone to Nodes in a Cluster](#)
- [Locating and Viewing Log Files Generated During Cloning](#)

Introduction to Cloning Oracle RAC

Cloning is the process of copying an existing Oracle RAC 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 Oracle 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 a Secure Shell (SSH) terminal session, if required.
- Cloning enables you to create an 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 patch sets 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 Oracle RAC by cloning is a very quick process. For example, cloning an Oracle 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).

The cloned installation behaves the same as the source installation. For example, the cloned Oracle home can be removed using Oracle Universal Installer 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 Oracle RAC

Use this overview to understand the procedures you use to clone Oracle RAC.

In the preparation phase, you create a copy of an Oracle home that you then use to perform the cloning procedure on one or more nodes. You also install Oracle Clusterware.

Install Oracle RAC

Use the detailed instructions in *Oracle Real Application Clusters Installation Guide* for your platform for your platform to install the Oracle RAC software and patches:

1. Install Oracle RAC and choose the **Software only** installation option.
2. Patch the release to the required level (for example, 12.1.0.n).
3. Apply one-off patches, if necessary.

Create a backup of the source home

Create a copy of the Oracle RAC home. Use this file to copy the Oracle RAC home to each node in the cluster.

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/12c/db_1
# tar -zcvf /pathname/db1120.tgz .
```

Install and start Oracle Clusterware

Before you can use cloning to create an Oracle RAC home, you must first install and start Oracle Clusterware on the node or nodes to which you want to copy a cloned Oracle RAC home. In other words, you configure an Oracle RAC home that you cloned from a source cluster onto the nodes in a target cluster in the same order that you installed the Oracle Clusterware and Oracle RAC software components on the original nodes.

Deploying Oracle RAC Clone to Nodes in a Cluster

After you complete the prerequisite tasks described in "Preparing to Clone Oracle RAC", you can deploy cloned Oracle homes.

Deploy the Oracle RAC database home to a cluster, as follows:

1. Perform any Oracle RAC preinstallation tasks, as described in your platform-specific Oracle RAC installation guide, to prepare the new cluster nodes, such things as:
 - Specify the kernel parameters.
 - Ensure Oracle Clusterware is active.
 - Ensure that Oracle ASM is active and that at least one Oracle ASM disk group exists and is mounted.
2. Deploy the Oracle RAC database software, as follows:
 - a. Copy the clone of the Oracle home to all nodes. For example:

```
[root@node1 root]# mkdir -p /opt/oracle/product/12c/db
[root@node1 root]# cd /opt/oracle/product/12c/db
[root@node1 db]# tar -zxvf /path_name/db1120.tgz
```

When providing the home location and *path_name*, 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.

- b. If either the `oracle` user or the `oinstall` group, or both is different between the source and destination nodes, then change the ownership of the Oracle Inventory files, as follows:

```
[root@node1]# chown -R oracle:oinstall /opt/oracle/product/12c/db
```

When you run the preceding command on the Oracle RAC home, it clears `setuid` and `setgid` information from the Oracle binary.

 **Note:**

You can perform this step at the same time you perform Step 3 and Step 4 to run the `clone.pl` and `$ORACLE_HOME/root.sh` scripts on each cluster node.

3. Run the `clone.pl` script on each node, which performs the main Oracle RAC cloning tasks, as follows:
 - a. Supply the environment variables and cloning parameters in the `start.sh` script, as described in Table 9-2 and Table 9-3. Because the `clone.pl` script is sensitive to the parameters being passed to it, you must be accurate in your use of brackets, single quotation marks, and double quotation marks.
 - b. Run the script as `oracle` or the user that owns the Oracle RAC software.

The following table lists and describes the `clone.pl` script parameters:

Table 9-1 clone.pl Script Parameters

Parameter	Description
<code>ORACLE_HOME=Oracle_home</code>	The complete path to the Oracle home you want to clone. If you specify an invalid path, then the script exits. This parameter is required.
<code>ORACLE_BASE=ORACLE_BASE</code>	The complete path to the Oracle base you want to clone. If you specify an invalid path, then the script exits. This parameter is required.
<code>ORACLE_HOME_NAME=Oracle_home_name</code> <code>-defaultHomeName</code>	The Oracle home name of the home you want to clone. Optionally, you can specify the <code>-defaultHomeName</code> flag. This parameter is optional.
<code>ORACLE_HOME_USER=Oracle_home_user</code>	The OracleHomeUser for Windows services. This parameter is applicable to Windows only and is optional.
<code>OSDBA_GROUP=group_name</code>	Specify the operating system group you want to use as the OSDBA privileged group. This parameter is optional.
<code>OSOPER_GROUP=group_name</code>	Specify the operating system group you want to use as the OSOPER privileged group. This parameter is optional.
<code>OSASM_GROUP=group_name</code>	Specify the operating system group you want to use as the OSASM privileged group. This parameter is optional.
<code>OSBACKUPDBA_GROUP=group_name</code>	Specify the operating system group you want to use as the OSBACKUPDBA privileged group. This parameter is optional.
<code>OSDGDBA_GROUP=group_name</code>	Specify the operating system group you want to use as the OSDGDBA privileged group. This parameter is optional.
<code>OSKMDBA_GROUP=group_name</code>	Specify the operating system group you want to use as the OSKMDBA privileged group. This parameter is optional.
<code>-debug</code>	Specify this option to run the <code>clone.pl</code> script in debug mode
<code>-help</code>	Specify this option to obtain help for the <code>clone.pl</code> script.

The following example shows an excerpt from the `start.sh` script that calls the `clone.pl` script:

```
ORACLE_BASE=/opt/oracle
ORACLE_HOME=/opt/oracle/product/12c/db
```

```
cd $ORACLE_HOME/clone
THISNODE='host_name'

E01=ORACLE_HOME=/opt/oracle/product/12c/db
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
```

The following table lists and describes the environment variables E01, E02, and E03 that are shown in bold typeface in the preceding example:

Table 9-2 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.

The following table lists and describes the cloning parameters C01 and C02, that are shown in bold typeface in the preceding example:

Table 9-3 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.

The following example shows an excerpt from the `start.bat` script that the user must create that calls the `clone.pl` script:

```
set ORACLE_home=C:\oracle\product\12c\db1
cd %ORACLE_home%\clone\bin
set THISNODE=%hostname%
set E01=ORACLE_HOME=%ORACLE_home%
set E02=ORACLE_HOME_NAME=OraDBRAC
set E03=ORACLE_BASE=Oracle_Base
set C01="CLUSTER_NODES={node1,node2}"
set C02="-O LOCAL_NODE=%THISNODE%"
perl clone.pl %E01% %E02% %E03% %C01% %C02%
```

4.

 **Note:**

This step applies to Linux and UNIX installations, only.

Run the `$ORACLE_HOME/root.sh` as the `root` operating system user as soon as the `clone.pl` procedure completes on the node.

```
[root@node1 root]# /opt/oracle/product/12c/db/root.sh -silent
```

Note that you can run the script on each node simultaneously:

```
[root@node2 root]# /opt/oracle/product/12c/db/root.sh -silent
```

Ensure the script has completed on each node before proceeding to the next step.

5.

 **Note:**

You need only run DBCA on one node in the cluster to create Oracle RAC instances on all nodes.

This step shows how to run DBCA in silent mode and provide response file input to create the Oracle RAC instances.

The following example creates an Oracle RAC database named ERI on each node, creates database instances on each node, registers the instances in OCR, creates the database files in the Oracle ASM disk group called `DATA`, and creates sample schemas. It also sets the `SYS`, `SYSTEM`, `SYSMAN` and `DBSNMP` passwords to `password`, which is the password for each account:

```
[oracle@node1 oracle]$ export ORACLE_HOME=/opt/oracle/product/12c/db
[oracle@node1 oracle]$ cd $ORACLE_HOME/bin/
[oracle@node1 bin]$ ./dbca -silent -createDatabase -templateName
General_Purpose.dbc \
-gdbName ERI -sid ERI \
-sysPassword password -systemPassword password \
-sysmanPassword password -dbsnmpPassword password \
-emConfiguration LOCAL \
-storageType ASM -diskGroupName DATA \
-datafileJarLocation $ORACLE_HOME/assistants/dbca/templates \
-nodelist node1,node2 -characterset WE8ISO8859P1 \
-obfuscatedPasswords false -sampleSchema true
```

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/cloneActiontimestamp.log`
Contains a detailed log of the actions that occur during the Oracle Universal Installer part of the cloning.
- `Central_Inventory/logs/oraInstalltimestamp.err`
Contains information about errors that occur when Oracle Universal Installer is running.
- `Central_Inventory/logs/oraInstalltimestamp.out`
Contains other miscellaneous messages generated by Oracle Universal Installer.

- `$ORACLE_HOME/clone/logs/clonetimestamp.log`
Contains a detailed log of the actions that occur before cloning and during the cloning operations.
- `$ORACLE_HOME/clone/logs/errortimestamp.log`
Contains information about errors that occur before cloning and during cloning operations.

Table 9-4 describes how to find the location of the Oracle inventory directory.

Table 9-4 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	<code>/var/opt/oracle/oraInst.loc</code>
IBM AIX and Linux	<code>/etc/oraInst.loc</code> file.
Windows	<code>C:\Program Files\Oracle\Inventory</code>

10

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 nodes in an existing cluster.

To add Oracle RAC to nodes in a new cluster, see [Cloning Oracle RAC to Nodes in a New Cluster](#).

This chapter contains the following topics:

- [About Adding Nodes Using Cloning in Oracle RAC Environments](#)
- [Cloning Local Oracle Homes on Linux and UNIX Systems](#)
- [Cloning Shared Oracle Homes on Linux and UNIX Systems](#)
- [Cloning Oracle Homes on Windows Systems](#)

About Adding Nodes Using Cloning in Oracle RAC 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

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](#)" for more information.

Cloning Local Oracle Homes on Linux and UNIX Systems

Add nodes to existing Oracle RAC environments by cloning a local (non-shared) Oracle home in Linux and UNIX system environments.

Complete the following steps to clone Oracle Database with Oracle RAC software:

1. Follow the steps in the "Preparing to Clone Oracle RAC" to create a copy of an Oracle home that you then use to perform the cloning procedure on one or more nodes.
2. Use the `tar` utility to create an archive of the Oracle home on the existing node and copy it to the new node. If the location of the Oracle home on the source node is `$ORACLE_HOME`, then you must use this same directory as the destination location on the new node.

3. On the new node, configure the environment variables `ORACLE_HOME` and `ORACLE_BASE`. Then 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=$ORACLE_BASE ORACLE_HOME=$ORACLE_HOME
ORACLE_HOME_NAME=Oracle_home_name -O -noConfig
```

4. Run the following command to run the configuration assistants to configure Oracle RAC on the new nodes:

```
$ORACLE_HOME/cfgtoollogs/configToolFailedCommands
```

This script contains all commands that failed, were skipped, or were canceled during the installation. You can use this script to run the database configuration assistants outside of Oracle Universal Installer. Note that before you run the script you should check the script to see if any passwords within it need to be updated.

5. Run the following command on the existing node from the `$ORACLE_HOME/oui/bin` directory to update the inventory in the Oracle Database home with Oracle RAC, specified by `Oracle_home`, 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 nodes:

```
./runInstaller -updateNodeList ORACLE_HOME=$ORACLE_HOME -O "CLUSTER_NODES={existing_node,new_node2,new_node3}"
```

6. On each new node, go to the `$ORACLE_HOME` directory and run the following command:

```
./root.sh
```

7. From the node that you cloned, run Database Configuration Assistant (DBCA) to add Oracle RAC database instances on the new nodes.

Cloning Shared Oracle Homes on Linux and UNIX Systems

Add nodes to existing Oracle RAC environments by cloning a shared Oracle home in Linux and UNIX system environments.

Complete the following steps to clone Oracle Database with Oracle RAC software:

1. Follow the steps in the "Preparing to Clone Oracle RAC" to create a copy of an Oracle home that you then use to perform the cloning procedure on one or more nodes.
2. On the new node, configure the environment variables `ORACLE_HOME` and `ORACLE_BASE`. Then 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, `Oracle_home_name` is the name of the Oracle home, and the `-cfs` option indicates the Oracle home is shared:

```
perl clone.pl -O 'CLUSTER_NODES={existing_node,new_node2,new_node3}'
-O LOCAL_NODE=new_node2 ORACLE_BASE=$ORACLE_BASE ORACLE_HOME=$ORACLE_HOME
ORACLE_HOME_NAME=Oracle_home_name [-cfs]
```

 **Note:**

In the preceding command:

- Use the `-cfs` option for a shared Oracle Database home with Oracle RAC.
- The value for the `ORACLE_HOME_NAME` parameter must be that of the node you are cloning.

3. Run the following command on the existing node from the `$ORACLE_HOME/oui/bin` directory to update the inventory in the Oracle Database home with Oracle RAC, specified by `Oracle_home`, 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 nodes:

```
./runInstaller -updateNodeList ORACLE_HOME=$ORACLE_HOME "CLUSTER_NODES={existing_node,new_node2,new_node3}"
```

4. On each new node, go to the `$ORACLE_HOME` directory and run the following command:

```
./root.sh
```

5. From the node that you cloned, run Database Configuration Assistant (DBCA) to add Oracle RAC database instances to the new nodes.

Cloning Oracle Homes on Windows Systems

Add nodes to existing Oracle RAC environments by cloning a shared or local Oracle home in Windows system environments.

Complete the following steps to clone Oracle Database with Oracle RAC software:

1. If you have a local Oracle home, then use the ZIP utility to create an archive of the Oracle Database home with Oracle RAC on the existing node and copy it to the new node. Otherwise, proceed to the next step.

Extract the Oracle Database with Oracle RAC home files from the ZIP file on the new node in the same directory in which the Oracle Database home with Oracle RAC resided on the existing node. For example, assume that the location of the destination Oracle RAC home on the new node is `%ORACLE_HOME%`.

2. On the new node, go to the `%ORACLE_HOME%\clone\bin` directory and run the following command, where `Oracle_Home` is the Oracle Database home, `Oracle_Home_Name` is the name of the Oracle Database home, `Oracle_Base` is the Oracle base directory, `user_name` is the name of the Oracle home user (a non-Administrator user) for the Oracle home being cloned, `existing_node` is the name of the existing node, and `new_node` is the name of the new node:

```
perl clone.pl ORACLE_HOME=Oracle_Home ORACLE_BASE=Oracle_Base
ORACLE_HOME_NAME=Oracle_Home_Name ORACLE_HOME_USER=user_name
-O 'CLUSTER_NODES={existing_node,new_node}'
-O LOCAL_NODE=new_node
```

If you have a shared Oracle Database home with Oracle RAC, then append the `-cfs` option to the command to indicate that the Oracle home is shared, as shown in the following example:

```
perl clone.pl ORACLE_HOME=Oracle_Home ORACLE_BASE=Oracle_Base  
ORACLE_HOME_NAME=Oracle_Home_Name ORACLE_HOME_USER=user_name  
-O 'CLUSTER_NODES={existing_node,new_node}' -O LOCAL_NODE=new_node  
[-cfs -noConfig]
```

 **Note:**

- The `ORACLE_HOME_USER` is required *only* if you are cloning a secured Oracle home.
- Use the `-cfs` and `-noConfig` options for a shared Oracle Database home with Oracle RAC.
- The value for the `ORACLE_HOME_NAME` parameter must be that of the node you are cloning. To obtain the `ORACLE_HOME_NAME`, look in the registry on the node you cloning for the `ORACLE_HOME_NAME` parameter key under `HKEY_LOCAL_MACHINE\SOFTWARE\oracle\KEY_OraCRs12c_home1`.

3. On the existing node, from the `%ORACLE_HOME%\oui\bin` directory run the following command to update the inventory in the Oracle Database home with Oracle RAC, specified by `Oracle_home`, where `existing_node` is the name of the existing node, and `new_node` is the name of the new node:

```
setup.exe -updateNodeList ORACLE_HOME=Oracle_home "CLUSTER_NODES=  
{existing_node,new_node}" LOCAL_NODE=existing_node
```

4. From the node that you cloned, run DBCA to add Oracle RAC database instances to the new nodes.

11

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

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.

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 "Cloning Oracle RAC to Nodes in a New Cluster".

The topics in this chapter include the following:

- [Adding Oracle RAC to Nodes with Oracle Clusterware Installed](#)
- [Deleting Oracle RAC from a Cluster Node](#)

Note:

- Ensure that you have a current backup of Oracle Cluster Registry (OCR) before adding or deleting Oracle RAC by running the `ocrconfig -showbackup` command.
- The phrase "target node" as used in this chapter refers to the node to which you plan to extend the Oracle RAC environment.

Adding Oracle RAC to Nodes with Oracle Clusterware Installed

Before beginning this procedure, ensure that your existing nodes have the correct path to the *Grid_home* and that the `$ORACLE_HOME` environment variable is set to the Oracle RAC home.

- If you are using a local (non-shared) Oracle home, then you must extend the Oracle RAC database home that is on an existing node (`node1` in this procedure) to a target node (`node3` in this procedure).

Navigate to the `Oracle_home/addnode` directory on `node1` and run the `addnode.sh` script.

If you want to perform a silent installation, run the `addnode.sh` script using the following syntax:

```
$ ./addnode.sh -silent "CLUSTER_NEW_NODES={node3}"
```

- If you have a shared Oracle home that is shared using Oracle Automatic Storage Management Cluster File System (Oracle ACFS), then do the following to extend the Oracle database home to `node3`:

1. Start the Oracle ACFS resource on the new node by running the following command as `root` from the `Grid_home/bin` directory:

```
# srvctl start filesystem -device volume_device [-node node_name]
```

 **Note:**

Make sure the Oracle ACFS resources, including Oracle ACFS registry resource and Oracle ACFS file system resource where the Oracle home is located, are online on the newly added node.

2. Run the following command as the user that installed Oracle RAC from the `Oracle_home/oui/bin` directory on the node you are adding to add the Oracle RAC database home:

```
$ ./runInstaller -attachHome ORACLE_HOME="ORACLE_HOME"
"CLUSTER_NODES={node3}"
LOCAL_NODE="node3" ORACLE_HOME_NAME="home_name" -cfs
```

3. Navigate to the `Oracle_home/addnode` directory on `node1` and run the `addnode.sh` script as the user that installed Oracle RAC using the following syntax:

```
$ ./addnode.sh -noCopy "CLUSTER_NEW_NODES={node3}"
```

 **Note:**

Use the `-noCopy` option because the Oracle home on the destination node is already fully populated with software.

- If you have a shared Oracle home on a shared file system that *is not* Oracle ACFS, then you must first create a mount point for the Oracle RAC database home on the target node, mount and attach the Oracle RAC database home, and update the Oracle Inventory, as follows:

1. Run the `srvctl config database -db db_name` command on an existing node in the cluster to obtain the mount point information.

2. Run the following command as `root` on `node3` to create the mount point:

```
# mkdir -p mount_point_path
```

3. Mount the file system that hosts the Oracle RAC database home.

4. Run the following command as the user that installed Oracle RAC from the `Oracle_home/oui/bin` directory on the node you are adding to add the Oracle RAC database home:

```
$ ./runInstaller -attachHome ORACLE_HOME="ORACLE_HOME" "CLUSTER_NODES=
{local_node_name}" LOCAL_NODE="node_name" ORACLE_HOME_NAME="home_name"
```

5. Update the Oracle Inventory as the user that installed Oracle RAC, as follows:

```
$ ./runInstaller -updateNodeList ORACLE_HOME=mount_point_path "CLUSTER_NODES=
{node_list}"
```

In the preceding command, `node_list` refers to a list of all nodes where the Oracle RAC database home is installed, including the node you are adding.

Run the `Oracle_home/root.sh` script on node3 as root.

 **Note:**

Oracle recommends that you back up the OCR after you complete the node addition process.

You can now add an Oracle RAC database instance to the target node using either of the procedures in the following sections.

- [Adding Policy-Managed Oracle RAC Database Instances to Target Nodes](#)
- [Adding Administrator-Managed Oracle RAC Database Instances to Target Nodes](#)

Adding Policy-Managed Oracle RAC Database Instances to Target Nodes

You must manually add undo and redo logs, unless you store your policy-managed database on Oracle Automatic Storage Management (Oracle ASM) and [Oracle Managed Files](#) is enabled.

If there is space in a server pool to add a node and the database has been started at least once, then Oracle Clusterware adds the Oracle RAC database instance to the newly added node and no further action is necessary.

 **Note:**

The database must have been started at least once before you can add the database instance to the newly added node.

If there is no space in any server pool, then the newly added node moves into the Free server pool. Use the `srvctl modify srvpool` command to increase the cardinality of a server pool to accommodate the newly added node, after which the node moves out of the Free server pool and into the modified server pool, and Oracle Clusterware adds the Oracle RAC database instance to the node.

Adding Administrator-Managed Oracle RAC Database Instances to Target Nodes

 **Note:**

The procedures in this section only apply to administrator-managed databases. Policy-managed databases use nodes when the nodes are available in the database's server pool.

You can use either Oracle Enterprise Manager or DBCA to add Oracle RAC database instances to the target nodes. To add a database instance to a target node with Oracle 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.

These tools guide you through the following tasks:

- Creating a new database instance on each target node
- Creating and configuring high availability components
- Creating the Oracle Net configuration for a non-default listener from the Oracle home
- 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 "Workload Management with Dynamic Database Services".

Using DBCA in Interactive Mode to Add 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 to the Oracle RAC home.
2. Start DBCA by entering `dbca` at the system prompt from the `Oracle_home/bin` directory.

DBCA performs certain CVU checks while running. However, you can also run CVU from the command line to perform various verifications.

DBCA displays the Welcome page for Oracle RAC. Click **Help** on any DBCA page for additional information.
3. Select **Instance Management**, click **Next**, and DBCA displays the Instance Management page.
4. Select **Add Instance** and click **Next**. DBCA displays the List of Cluster Databases page that shows the databases and their current status, such as `ACTIVE` or `INACTIVE`.
5. From the List of Cluster Databases page, select the active Oracle RAC database to which you want to add an instance. 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.
6. Click **Next** to add a new instance and DBCA displays the Adding an Instance page.
7. 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.

8. Review the information on the Summary dialog and click **OK** or click **Cancel** to end the instance addition operation. DBCA displays a progress dialog showing DBCA performing the instance addition operation.
9. 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 *node_list* consists of the names of the nodes on which you added database instances:

```
cluvfy comp admprv -o db_config -d Oracle_home -n node_list [-verbose]
```

10. Perform any necessary service configuration procedures, as described in "Workload Management with Dynamic Database Services".

Using DBCA in Silent Mode to Add Database Instances to Target Nodes

You can use DBCA in silent mode to add instances to nodes on which you have extended an Oracle Clusterware home and an Oracle Database home.

Before you run the `dbca` command, ensure that you have set the `ORACLE_HOME` environment variable correctly on the existing nodes. Run DBCA, supplying values for the variables described in [Table 11-1](#), as follows:

```
dbca -silent -nodeList node_name -gdbName gdb_name
[-instanceName instance_name -sysDBAUserName sysdba -sysDBAPassword
password]
```

Table 11-1 Variables in the DBCA Silent Mode Syntax

Variable	Description
<i>node_name</i>	The node on which you want to add (or delete) the instance.
<i>gdb_name</i>	Global database name.
<i>instance_name</i>	Name of the instance. Provide an instance name only if you want to override the Oracle naming convention for Oracle RAC instance names.
<i>sysdba</i>	Name of the Oracle user with <code>SYSDBA</code> privileges.
<i>password</i>	Password for the <code>SYSDBA</code> user.

Perform any necessary service configuration procedures, as described in "Workload Management with Dynamic Database Services".

Deleting Oracle RAC from a Cluster Node

To remove Oracle RAC from a cluster node, you must delete the database instance and the Oracle RAC software before removing the node from the cluster.

Note:

If there are no database instances on the node you want to delete, then proceed to "Removing Oracle RAC".

This section includes the following procedures to delete nodes from clusters in an Oracle RAC environment:

- [Deleting Instances from Oracle RAC Databases](#)
- [Removing Oracle RAC](#)
- [Deleting Nodes from the Cluster](#)

Deleting Instances from Oracle RAC Databases

The procedures for deleting database instances are different for policy-managed and administrator-managed databases.

Deleting a policy-managed database instance involves reducing the number of servers in the server pool in which the database instance resides. Deleting an administrator-managed database instance involves using DBCA to delete the database instance.

Deleting Policy-Managed Databases

To delete a policy-managed database, reduce the number of servers in the server pool in which a database instance resides by relocating the server on which the database instance resides to another server pool. This effectively removes the instance without having to remove the Oracle RAC software from the node or the node from the cluster.

For example, you can delete a policy-managed database by running the following commands on any node in the cluster:

```
$ srvctl stop instance -db db_unique_name -node node_name  
$ srvctl relocate server -servers "server_name_list" -serverpool Free
```

The first command stops the database instance on a particular node and the second command moves the node out of its current server pool and into the Free server pool.

Deleting Instances from Administrator-Managed Databases



Note:

Before deleting an instance from an Oracle RAC database using SRVCTL to do the following:

- If you have services configured, then relocate the services
- Modify the services so that each service can run on one of the remaining instances
- Ensure that the instance to be removed from an administrator-managed database is neither a preferred nor an available instance of any service

Using DBCA in Interactive Mode to Delete Instances from Nodes

The procedure in this section explains how to use DBCA in interactive mode to delete an instance from an Oracle RAC database.

To delete an instance using DBCA in interactive mode, perform the following steps:

1. 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 be running during this step.

2. On the DBCA Operations page, select **Instance Management** and click **Next**. DBCA displays the Instance Management page.
3. On the DBCA Instance Management page, select the instance to be deleted, select **Delete Instance**, and click **Next**.
4. 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. 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.

Click **No** and exit DBCA or click **Yes** to perform another operation. If you click **Yes**, then DBCA displays the Operations page.
5. Verify that the dropped instance's redo thread has been removed by using SQL*Plus on an existing node to query the `GV$LOG` view. If the redo thread is not disabled, then disable the thread. For example:

```
SQL> ALTER DATABASE DISABLE THREAD 2;
```

6. Verify that the instance has been removed from OCR by running the following command, where `db_unique_name` is the database unique name for your Oracle RAC database:

```
$ srvctl config database -db db_unique_name
```

7. 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.

Run the following command, where the variables are the same as those shown in [Table 11-1](#) 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 as shown in the following example where `password` is the password:

```
dbca -silent -deleteInstance [-nodeList node_name] -gdbName gdb_name  
-instanceName instance_name [-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.

Removing Oracle RAC

This procedure removes Oracle RAC software from the node you are deleting from the cluster and updates inventories on the remaining nodes.

1. If there is a listener in the Oracle RAC home on the node you are deleting, then you must disable and stop it before deleting the Oracle RAC software. Run the following commands on any node in the cluster, specifying the name of the listener and the name of the node you are deleting:

```
$ srvctl disable listener -l listener_name -n name_of_node_to_delete  
$ srvctl stop listener -l listener_name -n name_of_node_to_delete
```

2. Deinstall the Oracle home—only if the Oracle home is *not* shared—from the node that you are deleting by running the following command from the *Oracle_home* \deinstall directory:

```
deinstall -local
```

Caution:

If the Oracle home is shared, then *do not* run this command because it will remove the shared software. Proceed to the next step, instead.

Deleting Nodes from the Cluster

After you delete the database instance and the Oracle RAC software, 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 then you run scripts on the remaining nodes to update the node list.

12

Adding and Deleting Oracle RAC from Nodes on Windows Systems

This chapter describes how 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 Windows systems.

Note:

In this chapter, the entries for *Grid_home* refer to the full path name for the Oracle Grid Infrastructure home, and the entries for *Oracle_home* refer to substitutes for environment variables for the Oracle home with Oracle RAC.

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 "Cloning Oracle RAC to Nodes in a New Cluster".

This chapter includes the following topics:

- [Adding Oracle RAC to Nodes with Oracle Clusterware Installed](#)
- [Deleting Oracle RAC from a Cluster Node](#)

Note:

- Ensure that you have a current backup of Oracle Cluster Registry (OCR) before adding or deleting Oracle RAC by running the `ocrconfig -showbackup` command.
- For all of the add node and delete node procedures, temporary directories such as `%TEMP%` or `C:\Temp` *should not be* shared directories. If your temporary directories are shared, then set your temporary environment variable, such as `%TEMP%`, to a location on a local node. In addition, use a directory path that exists on all of the nodes.

Adding Oracle RAC to Nodes with Oracle Clusterware Installed

Before beginning this procedure, ensure that your existing nodes have the correct path to the *Grid_home* and that the *Oracle_home* environment variables are set correctly.

To add Oracle RAC database instances to nodes that already have Oracle Clusterware installed, you must extend the Oracle RAC home that is on an existing node (`node1` in this procedure) of the cluster to the target nodes.

1. Navigate to the `Oracle_home\addnode` directory on `node1` and run the `addnode.bat` script using the following syntax, where `node2` is the name of the node you are adding:

```
addnode.bat "CLUSTER_NEW_NODES={node2}"
```

To run this command in silent mode:

```
addNode.bat -silent "CLUSTER_NEW_NODES={node2}"
```

For the Oracle home directory you use, if an Oracle home user was specified when the Oracle Database software was installed, then OUI requires the password for the Oracle home user. OUI checks the wallet (stored in the OCR) for the user and extracts the password from there. If the user information is not contained in the wallet, then the `addnode.bat` script generates an error unless you specify the `-promptPasswd` flag on the command line.

2. If you store your policy-managed database on Oracle Automatic Storage Management (Oracle ASM), [Oracle Managed Files](#) is enabled, and if there is space in a server pool for `node2`, then `crsd` adds the Oracle RAC database instance to `node2` and no further action is necessary. If Oracle Managed Files is not enabled, then you must manually add undo and redo logs.

If there is no space in a server pool, then `node2` moves into the Free server pool. Use the `srvctl modify srvpool` command to increase the cardinality of the server pool to accommodate `node2`, after which time `node2` moves out of the Free server pool and into the modified server pool, and `crsd` adds the Oracle RAC database instance to `node2`.

3. If you have an administrator-managed database, then add a new instance on `node2`

If you have a shared Oracle home that is shared using Oracle Automatic Storage Management Cluster File System (Oracle ACFS), then do the following to extend the Oracle database home to `node2`:

1. Start the Oracle ACFS resource on the new node by running the following command as `root` from the `Grid_home\bin` directory:

```
$ srvctl start filesystem -device volume_device_name [-node node_name]
```

Note:

Make sure the Oracle ACFS resources, including Oracle ACFS registry resource and Oracle ACFS file system resource where the Oracle home is located, are online on the newly added node.

2. Run the following command as the user that installed Oracle RAC from the `Oracle_home\oui\bin` directory on the node you are adding to add the Oracle RAC database home:

```
setup.exe -attachHome ORACLE_HOME="ORACLE_HOME" LOCAL_NODE="node2"
ORACLE_HOME_NAME="home_name" -cfs
```

3. Navigate to the `Oracle_home\addnode` directory on `node1` and run the `addnode.bat` script as the user that installed Oracle RAC using the following syntax:

```
addnode.bat -noCopy "CLUSTER_NEW_NODES={node2}"
```

 **Note:**

Use the `-noCopy` option because the Oracle home on the destination node is already fully populated with software.

If you have a shared Oracle home on a shared file system that *is not* Oracle ACFS, then you must first create a mount point for the Oracle RAC database home on the target node, mount and attach the Oracle RAC database home, and update the Oracle Inventory, as follows:

1. Run the `srvctl config database -db db_name` command on an existing node in the cluster to obtain the mount point information.
2. Mount the file system that hosts the Oracle RAC database home.
3. Run the following command as the user that installed Oracle RAC from the `Oracle_home\oui\bin` directory on the node you are adding to add the Oracle RAC database home:

```
setup.exe -attachHome ORACLE_HOME="ORACLE_HOME" "CLUSTER_NODES=
  local_node_name}" LOCAL_NODE="node_name" ORACLE_HOME_NAME="home_name"
```

4. Update the Oracle Inventory as the user that installed Oracle RAC, as follows:

```
setup.exe -updateNodeList ORACLE_HOME=mount_point_path
"CLUSTER_NODES={node_list}"
```

In the preceding command, `node_list` refers to a list of all nodes where the Oracle RAC database home is installed, including the node you are adding.

 **Note:**

Oracle recommends that you back up your voting disk and Oracle Cluster Registry (OCR) files after you complete the node addition process.

Adding Administrator-Managed Oracle RAC Database Instances to Target Nodes

You can use either Oracle Enterprise Manager or DBCA to add Oracle RAC database instances to the target nodes.

This section describes using DBCA to add Oracle RAC database instances.

These tools guide you through the following tasks:

- Creating a new database instance on each target node
- Creating and configuring high availability components

- Creating the Oracle Net configuration for a non-default listener from the Oracle home
- 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.

Using DBCA in Interactive Mode to Add Database Instances to Target Nodes

To add a database instance to a target node using DBCA in interactive mode, perform the following steps:

1. Ensure that your existing nodes have the Oracle home environment variable set correctly.
2. Start DBCA by entering `dbca` at the system prompt from the `Oracle_home\bin` directory on an existing node.

DBCA performs certain CVU checks while running. However, you can also run CVU from the command line to perform various verifications.
3. On the Database Operations page, select **Instance Management**, click **Next**, and DBCA displays the Instance Management page.
4. Select **Add Instance** and click **Next**. DBCA displays the List of Cluster Databases page that shows the databases and their current status, such as `ACTIVE` or `INACTIVE`.
5. From the List of Cluster Databases page, select the active Oracle RAC database to which you want to add an instance. 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.
6. Click **Next** to add a new instance and DBCA displays the Adding an Instance page.
7. 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 new node name from the list.

Note:

If you installed the Oracle home with the Oracle Home User option, then DBCA prompts you for that password on this page.

8. Review the information on the Summary Page and click **Finish** to initiate instance addition operation. DBCA displays a progress dialog showing DBCA performing the instance addition operation.

Creating the OraMTS Service for Microsoft Transaction Server

Oracle Services for Microsoft Transaction Server (OraMTS) permit Oracle databases to be used as resource managers in Microsoft application-coordinated transactions. OraMTS acts as a proxy for the Oracle database to the Microsoft Distributed Transaction Coordinator (MSDTC). As a result, OraMTS provides client-side

connection pooling and allows client components that leverage Oracle to participate in promotable and distributed transactions. In addition, OraMTS can operate with Oracle databases running on any operating system, given that the services themselves are run on Windows.

On releases earlier than Oracle Database 12c, the OraMTS service was created as part of a software-only installation. Starting with Oracle Database 12c, you must use a configuration tool to create this service.

Create the OraMTS service after adding a node or performing a software-only installation for Oracle RAC, as follows:

1. Open a command window.
2. Change directories to %ORACLE_HOME%\bin.
3. Run the OraMTSctl utility to create the OraMTS Service, where *host_name* is a list of nodes on which the service should be created:

```
C:\.bin> oramtsctl.exe -new -host host_name
```

Using DBCA in Silent Mode to Add Database Instances to Target Nodes

Add instances to nodes on which you have extended an Oracle Clusterware home and an Oracle Database home.

Use DBCA in silent mode with the following syntax:

```
dbca -silent -addInstance -nodeList node_name -gdbName gdb_name  
[-instanceName instance_name -sysDBAUserName sysdba -sysDBAPassword password]
```

See Also:

[Table 11-1](#) for definitions of the variables used in the preceding syntax

Perform any necessary service configuration procedures.

Deleting Oracle RAC from a Cluster Node

To remove Oracle RAC from a cluster node, you must delete the database instance and the Oracle RAC software before removing the node from the cluster.

Note:

If there are no database instances on the node you want to delete, then remove Oracle RAC.

This section includes the following procedures to delete nodes from clusters in an Oracle RAC environment:

- [Deleting Instances from Oracle RAC Databases](#)
- [Removing Oracle RAC](#)

- [Deleting Nodes from the Cluster](#)

Deleting Instances from Oracle RAC Databases

The procedures for deleting instances are different for policy-managed and administrator-managed databases.

Deleting a policy-managed database instance involves reducing the size of the server pool in which the database instance resides. Deleting an administrator-managed database instance involves using DBCA to delete the database instance.

Deleting Policy-Managed Databases

To delete a policy-managed database, decrease the size of the server pool in which a database instance resides. This effectively removes the instance without having to remove the Oracle RAC software from the node or the node from the cluster.

For example, you can delete a policy-managed database by running the following commands on any node in the cluster:

```
$ srvctl stop instance -db db_unique_name -node node_name  
$ srvctl relocate server -servers "server_name_list" -serverpool Free
```

The first command stops on the instance on a particular node and the second command moves the list of servers out of their current server pool and into the Free server pool.

Deleting Instances from Administrator-Managed Databases



Note:

Before deleting an instance from an Oracle RAC database using SRVCTL, do the following:

- If you have services configured, then relocate the services
- Modify the services so that each service can run on one of the remaining instances
- Ensure that the instance to be removed from an administrator-managed database is neither a preferred nor an available instance of any service

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 OCR.

Run 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 Operations page, select **Instance Management**, click **Next**, and DBCA displays the Instance Management page.
4. On the Instance Management page, select **Delete Instance**, click **Next**, and DBCA displays the List of Cluster Databases page.
5. Select an Oracle RAC database from which to delete an instance. Click **Next** and DBCA displays the List of Cluster Database Instances page. The List of Cluster Database Instances page shows the instances that are associated with the Oracle RAC database that you selected and the status of each instance.
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. Click **Finish**.
 - b. Click **OK** on the Confirmation dialog to proceed to delete the instance.
 - c. Click **OK** on the next Confirmation dialog to delete the instance and related Optimal Flexible Architecture (OFA) directory structure.

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.

Click **No** and exit DBCA or click **Yes** to perform another operation. If you click **Yes**, then DBCA displays the Operations page.
7. Verify that the dropped instance's redo thread has been removed using SQL*Plus to query the `V$LOG` view from an existing instance. If the redo thread is not disabled, then disable the thread. For example:


```
SQL> ALTER DATABASE DISABLE THREAD 2;
```
8. Verify that the instance has been removed from OCR by running the following command, where `db_unique_name` is the name of the database:


```
srvctl config database -db db_unique_name
```
9. 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.

Run the following command, where the variables are the same as those shown in [Table 11-1](#) 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 as shown in the following example where `password` is the SYSDBA password:

```
dbca -silent -deleteInstance [-nodeList node_name] -gdbName gdb_name
-instanceName instance_name [-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.

Removing Oracle RAC

This procedure removes the Oracle RAC software from the node you are deleting from the cluster and updates inventories on the remaining nodes.

1. If there is a listener in the Oracle RAC home on the node you are deleting, then you must disable and stop it before deleting the Oracle RAC software. Run the following commands on any node in the cluster, specifying the name of the listener and the name of the node you are deleting:

```
C:\srvctl disable listener -listener listener_name -node name_of_node_to_delete  
C:\srvctl stop listener -listener listener_name -node name_of_node_to_delete
```

2. Deinstall the Oracle home from the node that you are deleting by running the following command from the *Oracle_home\deinstall* directory:

```
deinstall -local
```

If you have a shared Oracle RAC home, then append the `-cfs` option to the command example in this step and provide a complete path location for the cluster file system.

Deleting Nodes from the Cluster

After you delete 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.

13

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.

This chapter includes the following topics:

- [Deploying Oracle RAC for High Availability](#)
- [General Design Considerations for Oracle RAC](#)
- [General Database Deployment Topics for Oracle RAC](#)

Deploying Oracle RAC 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. 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 appears to be down to your users.

This section includes the following topics:

- [About Designing a High Availability System](#)
- [Best Practices for Deploying Oracle RAC in a High Availability Environment](#)
- [Consolidating Multiple Applications in a Database or Multiple Databases in a Cluster](#)
- [Scalability of Oracle RAC](#)

About Designing a High Availability System

For mission critical systems, you must be able to perform failover and recovery, and your environment must be resilient to all types of failures.

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 if one component fails the other hardware components can remain operational 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 it satisfies the service level requirements. Destructive testing 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 for 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.

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, you can:

- Remove TCP/IP timeout waits 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.
- Place voting disks on separate volume groups to mitigate outages due to slow I/O throughput. To survive the failure of x voting devices, configure $2x + 1$ mirrors.
- Use Oracle Database Quality of Service Management (Oracle Database QoS Management) to monitor your system and detect performance bottlenecks.
- Place OCR with I/O service times in the order of 2 milliseconds (ms) or less.
- Tune database recovery using the `FAST_START_MTTR_TARGET` initialization parameter.
- Use Oracle Automatic Storage Management (Oracle 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 Oracle Enterprise Manager to administer your entire Oracle RAC environment, not just the Oracle RAC database. Use Oracle Enterprise Manager to create and modify services, and to start and stop the cluster database instances and the cluster database.
- 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 start RMAN from Oracle Enterprise Manager Backup Manager or from the command line.
- If you use sequence numbers, then always use `CACHE` with the `NOORDER` option for optimal performance in sequence number generation. 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`. The `NOCACHE` and `ORDER` combination has the most negative effect on performance compared to other caching and ordering combinations.

 **Note:**

If your environment cannot tolerate sequence number gaps, then consider pre-generating the sequence numbers or use the `ORDER` and `CACHE` options.

- 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 Oracle ASM provides the infrastructure to manage multiple databases whether they are single instance databases or Oracle RAC databases.

Managing Capacity During Consolidation

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 the database—must be able to obtain CPU resources in a timely fashion and must be given higher priority in the system. Oracle Database Quality of Service Management (Oracle Database QoS Management) can assist consolidating multiple applications in a cluster or database by dynamically allocating CPU resources to meet performance objectives. You can also use cluster configuration policies to manage resources at the cluster level.

Managing the Global Cache Service Processes During Consolidation

Oracle recommends that the number of real time Global Cache Service Processes (LMS n) 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 that 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 LMS n created by the Oracle RAC instance. By default, Oracle Database 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 few busy LMS n processes. Oracle Database calculates the number of processes when the instance is started, and you must restart the instance to change the value.

Using a Database Cloud for Consolidation

A database cloud is a set of databases integrated by the Global Data Services framework into a single virtual server that offers one or more global services, while ensuring high performance, availability and optimal utilization of resources.

Global Data Services manages these virtualized resources with minimum administration overhead, and allows the database cloud to quickly scale to handle additional client requests. The databases that constitute a cloud can be globally distributed, and clients can connect to the database cloud by simply specifying a service name, without needing to know anything about the components and topology of the cloud.

A database cloud can be comprised of multiple database pools. A [database pool](#) is a set of databases within a database cloud that provide a unique set of global services and belong to a certain administrative domain. Partitioning of cloud databases into multiple pools simplifies service management and provides higher security by allowing each pool to be administered by a different administrator. A database cloud can span multiple geographic regions. A [region](#) is a logical boundary that contains database clients and servers that are considered to be close to each other. Usually a region corresponds to a data center, but multiple data centers can be in the same region if the network latencies between them satisfy the service-level agreements of the applications accessing these data centers.

Global services enable you to integrate locally and globally distributed, loosely coupled, heterogeneous databases into a scalable and highly available private database cloud. This database cloud can be shared by clients around the globe. Using

a private database cloud provides optimal utilization of available resources and simplifies the provisioning of database services.

Scalability of Oracle RAC

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 Database but the servers do not have to have 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 or 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.

General Design Considerations for Oracle RAC

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 use hash partitioning for 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.

General Database Deployment Topics for Oracle RAC

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 noncluster design, then your application will run well on an Oracle RAC database.

This section includes the following topics:

- [Tablespace Use in Oracle RAC](#)
- [Object Creation and Performance in Oracle RAC](#)
- [Node Addition and Deletion and the SYSAUX Tablespace in Oracle RAC](#)
- [Distributed Transactions and Oracle RAC](#)
- [Deploying OLTP Applications in Oracle RAC](#)
- [Flexible Implementation with Cache Fusion](#)
- [Deploying Data Warehouse Applications with Oracle RAC](#)
- [Data Security Considerations in Oracle RAC](#)

Tablespace Use in Oracle RAC

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 RAC

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 noncluster 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 RAC installation guide for guidelines about sizing the `SYSAUX` tablespace for multiple instances

Distributed Transactions and Oracle RAC

If you are running XA Transactions in an Oracle RAC environment and the performance is poor, then direct all branches of a tightly coupled distributed transaction to the same instance by creating 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.

Deploying OLTP Applications in Oracle RAC

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 if there are failures
- Scalability to accommodate increased system demands
- Load balancing according to demand fluctuations

The high availability features of Oracle Database 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 Database 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 RAC

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.

This section includes the following topics:

- [Speed-Up for Data Warehouse Applications on Oracle RAC](#)
- [Parallel Execution in Data Warehouse Systems and Oracle RAC](#)

Speed-Up for Data Warehouse Applications on Oracle RAC

Oracle RAC is ideal for data warehouse applications because it augments the noncluster benefits of Oracle Database. 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 available for the operation. This results in Oracle Database selecting the execution plans that are optimized for your Oracle RAC configuration.

Parallel Execution in Data Warehouse Systems and Oracle RAC

Parallel execution uses multiple processes to run SQL statements on one or more CPUs and is available on both noncluster 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 degree of parallelism assigned to each table or index.

Data Security Considerations in Oracle RAC

This section describes the following two Oracle RAC security considerations:

- [Transparent Data Encryption and Keystores](#)
- [Windows Firewall Considerations](#)

Transparent Data Encryption and Keystores

Oracle Database enables Oracle RAC nodes to share the [keystore](#) (wallet). This eliminates the need to manually copy and synchronize the keystore across all nodes. Oracle recommends that you create the keystore on a shared file system. This allows all instances to access the same shared keystore.

Oracle RAC uses keystores in the following ways:

1. Any keystore operation, like opening or closing the keystore, performed on any one Oracle RAC instance is applicable for all other Oracle RAC instances. This means that when you open and close the keystore for one instance, then it opens and closes the keystore for all Oracle RAC instances.
2. When using a shared file system, ensure that the `ENCRYPTION_WALLET_LOCATION` parameter for all Oracle RAC instances points to the same shared keystore location. The security administrator must also ensure security of the shared keystore by assigning appropriate directory permissions.

 **Note:**

If Oracle Automatic Storage Management Cluster File System (Oracle ACFS) is available for your operating system, then Oracle recommends that you store the keystore in Oracle ACFS. If you do not have Oracle ACFS in Oracle ASM, then use the Oracle ASM Configuration Assistant (ASMCA) to create it. You must add the mount point to the `sqlnet.ora` file in each instance, as follows:

```
ENCRYPTION_WALLET_LOCATION=
(SOURCE = (METHOD = FILE)
(METHOD_DATA =
(DIRECTORY = /opt/oracle/acfsmounts/data_keystore)))
```

This file system is mounted automatically when the instances start. Opening and closing the keystore, and commands to set or rekey and rotate the TDE master encryption key, are synchronized between all nodes.

3. A master key rekey performed on one instance is applicable for all instances. When a new Oracle RAC node comes up, it is aware of the current keystore open or close status.
4. Do not issue any keystore `ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN OR CLOSE` SQL statements while setting up or changing the master key.

Deployments where shared storage does not exist for the keystore require that each Oracle RAC node maintain a local keystore. After you create and provision a keystore on a single node, you must copy the keystore and make it available to all of the other nodes, as follows:

- For systems using Transparent Data Encryption with encrypted keystores, you can use any standard file transport protocol, though Oracle recommends using a secured file transport.
- For systems using Transparent Data Encryption with auto-login keystores, file transport through a secured channel is recommended.

To specify the directory in which the keystore must reside, set the `ENCRYPTION_WALLET_LOCATION` parameter in the `sqlnet.ora` file. The local copies of the keystore need not be synchronized for the duration of Transparent Data Encryption usage until the server key is re-keyed through the `ADMINISTER KEY MANAGEMENT SET KEY` SQL statement. Each time you issue the `ADMINISTER KEY MANAGEMENT SET KEY` statement on a database instance, you must again copy the keystore residing on that node and make it available to all of the other nodes. Then, you must close and reopen the keystore 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.

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.

14

Monitoring Performance

This chapter describes how to monitor and tune Oracle Real Application Clusters (Oracle RAC) performance.

This chapter includes the following topics:

- [Overview of Monitoring and Tuning Oracle RAC Databases](#)
- [Verifying the Interconnect Settings for Oracle RAC](#)
- [Influencing Interconnect Processing](#)
- [Performance Views in Oracle RAC](#)
- [Creating Oracle RAC Data Dictionary Views with CATCLUST.SQL](#)
- [Oracle RAC Performance Statistics](#)
- [Automatic Workload Repository in Oracle RAC Environments](#)
- [Active Session History Reports for Oracle RAC](#)
- [Monitoring Oracle RAC Statistics and Wait Events](#)

Overview of Monitoring and Tuning Oracle RAC Databases

This section includes the following topics:

- [Monitoring Oracle RAC and Oracle Clusterware](#)
- [Tuning Oracle RAC Databases](#)

Monitoring Oracle RAC and Oracle Clusterware

Using Oracle Enterprise Manager is the preferred method for monitoring Oracle RAC and Oracle Clusterware. 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, in addition to related hardware and software. For example, you can monitor your Oracle RAC database performance from your office, home, or a remote site, if you have access to a Web browser.

Oracle Enterprise Manager Cloud Control is cluster-aware and provides 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 and those issues 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.

Oracle Enterprise Manager accumulates data over specified periods of time, called collection-based data. Oracle Enterprise Manager also provides current data, called real-time data.

Oracle Database 2 Day + Real Application Clusters Guide provides complete information about monitoring performance with Oracle Enterprise Manager, including:

- Automatic Database Diagnostic Monitor and Oracle RAC Performance
- [The Cluster Database Home Page](#)
- [The Interconnects Page](#)
- [The Cluster Database Performance Page](#)

The Cluster Database Home Page

You can use Oracle Enterprise Manager with a client browser to 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 OCR, voting disk issues (if any), and node evictions

The Cluster Database Home page is similar to a noncluster 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, and links to all the database and Oracle Automatic Storage Management (Oracle 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 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 is also available as collections that have a historic view, which is useful 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 Database Performance Page

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.

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, Oracle 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.

Tuning Oracle RAC Databases

All noncluster tuning practices for Oracle Database apply to Oracle RAC databases.

Verifying the Interconnect Settings for Oracle RAC

Use SQL statements to verify the interconnect settings for Oracle RAC.

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 of the Oracle RAC database instance to which you are connected, query the `V$CLUSTER_INTERCONNECTS` and `V$CONFIGURED_INTERCONNECTS` views. For example:

Example 14-1 Verify Interconnect Settings with `V$CLUSTER_INTERCONNECTS`

```
SQL> SELECT * FROM V$CLUSTER_INTERCONNECTS;
```

NAME	IP_ADDRESS	IS_PUBLIC	SOURCE
eth2	10.137.20.181	NO	Oracle Cluster Repository

 **Note:**

You can query the `GV$CLUSTER_INTERCONNECTS` view to display the entries for all of the instances in the cluster.

Example 14-2 Verify Interconnect Settings with V\$CONFIGURED_INTERCONNECTS

```
SQL> SELECT * FROM V$CONFIGURED_INTERCONNECTS;
```

NAME	IP_ADDRESS	IS_PUBLIC	SOURCE
eth2	10.137.20.181	NO	Oracle Cluster Repository
eth0	10.137.8.225	YES	Oracle Cluster Repository

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 interprocess communication (IPC) buffer sizes.

The Oracle Cluster Registry (OCR) stores your system's interconnect information. Use the Oracle Interface Configuration (OIFCFG) command-line utility `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.

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.

 **Notes:**

- You can also use OIFCFG command to assign private network or private IP addresses.
- If your cluster has Oracle Clusterware 12c release 2 (12.2) or later installed, then you can assign either IPv4 or IPv6 addresses to multiple private networks. You must choose one or the other protocol and use that protocol for all of the private networks in the cluster.

Performance Views in Oracle RAC

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;
```

Creating Oracle RAC 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.

Oracle RAC Performance Statistics

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 RAC Environments

You can use Automatic Workload Repository to monitor performance statistics related to Oracle RAC databases.

[Automatic Workload Repository \(AWR\)](#) 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 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 RAC 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. 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.

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.

Monitoring Oracle RAC 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 (AWR), Statspack, or by ad-hoc queries of the dynamic performance views.

This section includes the following topics:

- [Oracle RAC Statistics and Events in AWR and Statspack Reports](#)
- [Oracle RAC 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](#)

Oracle RAC 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 RAC Wait Events

Analyzing and interpreting what causes sessions to wait 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 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

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 RAC](#)
- [Analyzing Performance Using GCS and GES Statistics](#)

Analyzing the Effect of Cache Fusion in Oracle RAC

The effect of accessing blocks in the global cache and maintaining coherency is represented by:

- The Global Cache Service (GCS) statistics for `current` and `cr` blocks, for example, `gc current blocks received`, `gc cr blocks received`, and so on
- The GCS 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 to guarantee [cache coherency](#) and Oracle RAC inherently does not cause any more I/O to disk than a nonclustered instance.

Analyzing Performance Using GCS and GES Statistics

You can monitor GCS performance by identifying data blocks and objects which are frequently used (*hot*) by all instances.

High concurrency on certain blocks may be identified by GCS 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 could be caused by any of the following:

- The blocks were pinned
- The blocks were held up by sessions
- The blocks were delayed by a log write on a remote instance

- A session on the same instance was already accessing a block which was in transition between instances and the current session needed to wait behind it (for example, `gc current block busy`)

Use the `V$SESSION_WAIT` view to identify objects and data blocks with contention. The GCS 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 having to query the `V$SESSION_WAIT` view 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.



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.

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

Describes how to monitor GCS 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 are usually 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 usually 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 in the preceding list 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.

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 frequent wait events that you should be aware of when interpreting performance data.

If user response times increase and a high proportion of time waited is for global cache, 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.

Wait events for Oracle RAC include the following categories:

- [Block-Related Wait Events](#)
- [Message-Related Wait Events](#)
- [Contention-Related Wait Events](#)
- [Load-Related Wait Events](#)

Block-Related Wait Events

The main wait events for block-related waits are:

- `gc current block 2-way`
- `gc current block 3-way`
- `gc cr block 2-way`
- `gc cr block 3-way`

The block-related 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.

Message-Related Wait Events

The main wait events for message-related waits are:

- `gc current grant 2-way`

- `gc cr grant 2-way`

The message-related 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).

Contention-Related Wait Events

The main wait events for contention-related waits are:

- `gc current block busy`
- `gc cr block busy`
- `gc buffer busy acquire/release`

The contention-related 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 `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 Database 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 Database 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 Database must queue these requests. The length of time that Oracle Database 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.

Load-Related Wait Events

The main wait events for load-related waits are:

- gc current block congested
- gc cr block congested

The load-related 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.

15

Converting Single-Instance Oracle Databases to Oracle RAC and Oracle RAC One Node

Procedures for converting from Oracle Database single-instance databases to Oracle Real Application Clusters (Oracle RAC) and Oracle RAC One Node databases.

The procedures in this chapter assume that your original single-instance database and the target Oracle RAC database are using the same release and running on the same platform. If you are upgrading from an earlier version of Oracle RAC to Oracle RAC 12c, then use Oracle Database Upgrade Assistant (DBUA).

This chapter includes the following topics:

- [Administrative Issues for Converting Databases to Oracle RAC](#)
- [Converting to Oracle RAC and Oracle RAC One Node Using DBCA](#)
- [Preparing to Convert with rconfig and Oracle Enterprise Manager](#)
- [Converting Databases to Oracle RAC Using rconfig](#)
- [Example of rconfig XML Input Files for ConvertToRAC](#)
- [Postconversion Steps](#)

Note:

You must use clustered Oracle Automatic Storage Management (Oracle ASM) instances for Oracle RAC databases.

Administrative Issues for Converting Databases to Oracle RAC

Note the following administrative considerations before converting single-instance databases to Oracle RAC:

- Backup procedures should be available before converting from a single-instance Oracle Database to Oracle RAC. This includes taking a backup of your existing database before converting to Oracle RAC and being prepared to backup your Oracle RAC database immediately following the conversion.
- For archiving with Oracle RAC environments, the archive file format requires a thread number.
- The archived logs from all instances of an Oracle RAC database are required for media recovery. Because of this requirement, if you archive to a file and you do not use a cluster file system, or some other means to provide shared file systems,

then you require a method of accessing the archive logs from all nodes on which the cluster database has instances.

- By default, all database files are migrated to Oracle Managed Files. This feature simplifies tablespace creation, ensures data file location consistency and compliance with Oracle Flexible Architecture rules, and reduces human error with data file management.

Converting to Oracle RAC and Oracle RAC One Node Using DBCA

You can use Database Configuration Assistant (DBCA) to convert from single-instance Oracle databases to Oracle RAC or Oracle RAC One Node databases.

DBCA automates the configuration of the control file attributes, creates the undo tablespaces and the redo logs, and creates the initialization parameter file entries for cluster-enabled environments. DBCA also configures Oracle Net Services, Oracle Clusterware resources, and the configuration for Oracle RAC database management using Oracle Enterprise Manager or the Server Control utility (SRVCTL).

Before you use DBCA to convert a single-instance database to an Oracle RAC or an Oracle RAC One Node database, ensure that your system meets the following conditions:

- Your system uses supported hardware and operating system software. Your system is configured properly to support an Oracle RAC database.
- The nodes have access to shared storage; for example, either Oracle Cluster File System or Oracle ASM is available and accessible from all nodes. On Linux on POWER systems, ensure that GPFS is available and accessible from all nodes.
- Your applications have no design characteristics that preclude their use with cluster database processing.

If your platform supports a cluster file system, then you can use it for Oracle RAC. You can also convert to Oracle RAC and use a non-shared file system. In either case, Oracle strongly recommends that you use Oracle Universal Installer to install Oracle Database 12c, which sets up the Oracle home and inventory in an identical location on each of the selected nodes in your cluster.

This section includes the following topics:

- [Converting Oracle Database Installations to Oracle RAC Using DBCA](#)
- [Converting Single Instance on a Cluster to Oracle RAC One Node Using DBCA](#)
- [Converting Single Instance on a Cluster to Oracle RAC Using DBCA](#)

Converting Oracle Database Installations to Oracle RAC Using DBCA

To convert from a single-instance Oracle Database that is on a non-clustered computer to Oracle RAC, perform the procedures described in the following sections, and in the order shown:

- [Use DBCA to Create an Image of the Single-Instance Database](#)
- [Complete the Oracle Clusterware Installation](#)
- [Validate the Cluster](#)

- [Copy the Preconfigured Database Image](#)
- [Install Oracle Database 12c Software with Oracle RAC](#)

Use DBCA to Create an Image of the Single-Instance Database

Use DBCA to create a preconfigured image of your single-instance database by using the following procedure:

1. Navigate to the `bin` directory in `$ORACLE_HOME`, and start DBCA.
2. At the Welcome page, click **Next**.
3. On the Operations page, select **Manage Templates**, and click **Next**.
4. On the Template Management page, select **Create a database template and From an existing database (structure as well as data)**, then click **Next**.
5. On the Source Database page, select the database name in the Database instance list, and click **Next**.
6. Use SQL to ensure that all pluggable databases (PDBs) are open, as follows:

```
SQL> SELECT name, open_mode FROM v$pdb;
```

If any of the PDBs are in a state other than OPEN, then open them using SQL.

7. On the Template Properties page, enter a name for your template in the Name field. Oracle recommends that you use the database name.

By default, the template files are generated in the directory `$ORACLE_HOME/assistants/dbca/templates`. You can enter a description of the file in the Description field, and change the template file location in the Template data file field.

When you have completed the entries, click **Next**.

8. On the Location of Database Related Files page, select **Maintain the file locations**, so that you can restore the database to the current directory structure, and click **Finish**.

DBCA generates two files: a database structure file (`template_name.dbc`), and a database preconfigured image file (`template_name.dfb`).

Complete the Oracle Clusterware Installation

Complete the installation of Oracle Clusterware.

Validate the Cluster

Validate the cluster configuration using Cluster Verification Utility (CVU).

Copy the Preconfigured Database Image

Copy the preconfigured database image. This includes copying the database structure `*.dbc` file and the database preconfigured image `*.dfb` file (that you used DBCA to create in a previous section) to a temporary location on the node in the cluster from which you plan to run DBCA.

Install Oracle Database 12c Software with Oracle RAC

1. Run Oracle Universal Installer to install an Oracle Database with Oracle RAC.
2. Select **Cluster Installation Mode** on the Specify Hardware Cluster Installation page of Oracle Universal Installer, and select the nodes to include in your Oracle RAC database.
3. On the Oracle Universal Installer Database Configuration Types page, select the **Advanced** installation type.

After installing the Oracle Database software, Oracle Universal Installer runs postinstallation configuration tools, such as Net Configuration Assistant (NETCA), DBCA, and so on.
4. On the DBCA Template Selection page, use the template that you copied to a temporary location in the previous section. Use the browse option to select the template location.

Select the option that you want to deploy. Your choices are the following: Oracle RAC database; Oracle RAC One Node database; or Oracle single-instance database.
5. After creating the Oracle RAC database, DBCA displays the Password Management page on which you must change the passwords for database users who have SYSDBA and SYSOPER privileges. When DBCA exits, the conversion process is complete.

Converting Single Instance on a Cluster to Oracle RAC One Node Using DBCA

Use DBCA to convert a single-instance Oracle Database to Oracle RAC One Node by using the following procedure:

1. Change directory to `$ORACLE_HOME/bin`.
2. Start DBCA:

```
$ dbca
```
3. From the Welcome window, select **Oracle RAC One Node** database.
4. Use the template that you selected to deploy in the previous section.

Converting Single Instance on a Cluster to Oracle RAC Using DBCA

There are three scenarios in which a single-instance database can exist on a cluster node:

- Scenario 1: The Oracle home for the single-instance database was installed on a cluster node and has Oracle RAC enabled.
- Scenario 2: The Oracle home for the single-instance database was installed on a cluster node, but the Oracle RAC feature is disabled for this Oracle home.
- Scenario 3: The Oracle home for the single-instance database was installed on only the local node in a cluster. This happens when you select the Local Installation option on the Oracle Universal Installer Specify Hardware Cluster Installation page during the Oracle Database 12c installation.

Single-Instance Database on a Cluster Running from an Oracle RAC-Enabled Home

Perform the following procedures to convert a single-instance database on a cluster node running from an Oracle home that has the Oracle RAC option enabled.

1. Use DBCA to create a preconfigured image of your single-instance database. To perform the conversion manually, shut down the single-instance database.
2. Add nodes to your cluster. Ensure that all nodes can access the shared storage used by Oracle Clusterware and Oracle RAC.
3. From the existing Oracle home, extend this home to the new nodes.
4. From a newly added node, configure the listeners on the additional nodes using NETCA. Choose the same port number and protocol that you used on the existing node. If NETCA displays the existing node in the node list page, then do not select this node, because the listener is already configured on it.
5. Convert the database using one of the following procedures:
 - [Automated Conversion Procedure Using DBCA](#)
 - [Manual Conversion Procedure](#)

Automated Conversion Procedure Using DBCA

If you used DBCA to create a preconfigured image of your single-instance database as described in a previous section, then perform the following steps to complete the conversion to an Oracle RAC database:

1. Start DBCA from the initial node. Select the names of the nodes to include as part of your cluster database. On the Template Selection page, select the preconfigured template that you created. Enter the database name and respond to the remaining DBCA prompts.
2. Specify the shared storage location for the Oracle Database data files.

After creating the Oracle RAC database, DBCA displays the Password Management page on which you must change the passwords for the database users who have SYSDBA and SYSOPER privileges. When DBCA exits, the conversion process is complete.

Manual Conversion Procedure

If you did not use DBCA to create a preconfigured image of your single-instance database as described in a previous section, then perform the following steps to complete the conversion:

1. Create the Optimal Flexible Architecture directory structure on each of the nodes that you have added.
2. Re-create the control files by running the `CREATE CONTROLFILE` SQL statement with the `REUSE` keyword and specify `MAXINSTANCES` and `MAXLOGFILES`, and so on, as needed for your Oracle RAC configuration. The `MAXINSTANCES` recommended default is 32.
3. Shut down the database instance.

4. If your single-instance database was using an SPFILE, then create a temporary parameter file (PFILE) from the SPFILE using the following SQL statement:

```
CREATE PFILE='pfile_name' from spfile='spfile_name'
```

5. Set the `CLUSTER_DATABASE` parameter to `TRUE`, and set the `INSTANCE_NUMBER` parameter to a unique value for each instance, using the `sid.parameter=value` syntax.

If you optimized memory usage on your single-instance database, then adjust the size of the system global area (SGA) to avoid swapping and paging when you convert to Oracle RAC. You should make this adjustment because Oracle RAC requires about 350 bytes for each buffer to accommodate the Global Cache Service (GCS). For example, if you have 10,000 buffers, then Oracle RAC requires approximately 350 multiplied by 10,000 bytes more memory. Therefore, adjust the size of the SGA by changing the `DB_CACHE_SIZE` and `DB_nK_CACHE_SIZE` parameters accordingly.

6. Start the database instance using the PFILE created in Step 4.
7. If your single-instance database was using automatic undo management, then create an undo tablespace for each additional instance using the `CREATE UNDO TABLESPACE` SQL statement.
8. Create redo threads that have at least two redo logs for each additional instance. Enable the new redo threads by using an `ALTER DATABASE` SQL statement. Then, shut down the database instance.
9. Copy the Oracle password file from the initial node, or from the node on which you are working, to the corresponding location on the additional nodes on which the cluster database will have an instance. Replace the `ORACLE_SID` name in each password file appropriately for each additional instance.
10. Set the `REMOTE_LISTENER` parameter to the single client access name (SCAN) and port.
11. Configure the net service entries for the database and instances, and address entries for the `LOCAL_LISTENER` for each instance and for the `REMOTE_LISTENER` in the `tnsnames.ora` file, and copy the `tnsnames.ora` file to all nodes.
12. Create the SPFILE from the PFILE.
13. Create the `$ORACLE_HOME/dbs/initsid.ora` file that contains the following entry, where `spfile_path_name` is the complete path name of the SPFILE:

```
spfile='spfile_path_name'
```

14. On the local node, use SQL*Plus to run `catclust.sql`. This script creates the dictionary views needed for Oracle RAC databases. For example:

```
SQL> start ?/rdbms/admin/catclust.sql
```

15. Add the configuration for the Oracle RAC or Oracle RAC One Node database and its instance-to-node mapping using `SRVCTL`.

- a. To add the configuration of an Oracle RAC database, use the following commands:

```
$ srvctl add database -dbname db_name -oraclehome Oracle_home -spfile
spfile_path_name
$ srvctl add instance -dbname db_name -instance inst1_name -node node1_name
$ srvctl add instance -dbname db_name -instance inst2_name -node node2_name
...
```

- b. To add the configuration of an Oracle RAC One Node database, use the following command:

```
$ srvctl add database -dbname db_name -dbtype RACONENODE -oraclehome  
Oracle_home  
-spfile spfile_path_name
```

16. Start the Oracle RAC or Oracle RAC One Node database using SRVCTL:

```
srvctl start database -d db_name
```

After starting the database with SRVCTL, your conversion process is complete. You can run the following SQL statement to see the status of all the instances in your Oracle RAC database:

```
SQL> SELECT * FROM v$instance;
```

Single-Instance Database on a Cluster Running from an Oracle RAC-Disabled Home

You can create a single-instance database on a cluster running from an Oracle home with the Oracle RAC option disabled. To create an Oracle home on a cluster with Oracle RAC disabled, you can select **local** and **non-cluster** on the Node Selection Page of Oracle Universal Installer when installing the Oracle Database software. You can also performed a one-node cluster (with Oracle RAC) installation, but later disable the Oracle RAC option.

Perform the following procedures to convert this type of single-instance database to an Oracle RAC or Oracle RAC One Node database:

1. Use DBCA to create a preconfigured image of your single-instance database as described in a previous section. To perform the conversion manually, shut down the single-instance database.
2. Change the directory to the `lib` subdirectory in the `rdbms` directory under the Oracle home.
3. Relink the `oracle` binary by running the following commands:

```
make -f ins_rdbms.mk rac_on  
make -f ins_rdbms.mk ioracle
```

4. Add nodes to your cluster. Ensure that all nodes can access the shared storage used by Oracle Clusterware and Oracle RAC.

Preparing to Convert with rconfig and Oracle Enterprise Manager

You can use `rconfig`, or Oracle Enterprise Manager to assist with converting a single-instance database installation to an Oracle RAC database. The first of these, `rconfig`, is a command-line utility. Oracle Enterprise Manager Cloud Control database administration option, Convert to Cluster Database, provides a GUI-based conversion tool. The following sections describe how to use these conversion tools:

- [Prerequisites for Converting to Oracle RAC Databases](#)
- [Configuration Changes During Oracle RAC Conversion Using rconfig](#)
- [Converting Databases to Oracle RAC Using rconfig or Oracle Enterprise Manager](#)

- [Converting Databases to Oracle RAC Using Oracle Enterprise Manager](#)

 **Note:**

Before you start the conversion, back up your existing database; you should take a backup of your database before starting any major change.

Prerequisites for Converting to Oracle RAC Databases

Before you convert a single-instance database to an Oracle RAC database, ensure that the following conditions are met for each cluster node that you intend to make an Oracle RAC database node:

- Oracle Clusterware 12c is installed, configured, and running.
- Oracle RAC 12c software is installed.
- The Oracle software has the Oracle RAC option enabled.
- Shared storage, either Oracle Cluster File System or Oracle ASM, is available and accessible from all nodes.
- User equivalence exists for the `oracle` account, or the user account used to install the Oracle software.
- If you intend to use Oracle Enterprise Manager, then the Oracle Management Agent on each node is configured and running, and is configured with cluster and host information.
- You have backed up your existing database.

 **Note:**

You must use clustered Oracle ASM instances for Oracle RAC databases.

Configuration Changes During Oracle RAC Conversion Using rconfig

The following changes occur when you convert a single-instance database to Oracle RAC using the `rconfig` utility:

- During the conversion, `rconfig` places the target Oracle RAC database into archive log mode, and enables archiving for the database. If you do not plan to use archive log space, then you can disable archive logging after the conversion has completed.
- For the **Shared Storage Type** value, if you enter **CFS**, and you use a cluster file system for your single-instance database storage, then `rconfig` converts the environment to use Oracle Managed Files for database storage, and places the data files in a subdirectory located under the shared storage location.
- During the conversion, `rconfig` moves database files to a specified shared location, and configures them using Oracle Managed Files.

To avoid using Oracle Managed Files with your converted database, the single-instance database files must be located in a shared file system, and you must indicate that `rconfig` should not move the files.

Converting Databases to Oracle RAC Using rconfig or Oracle Enterprise Manager

The following list describes scenarios for converting a single-instance Oracle database to an Oracle RAC database:

- Converting a single-instance Oracle Database 12c database to an Oracle RAC 12c database, running out of the *same* Oracle home and using the same data files as the single-instance database.

In this scenario, run the `rconfig` utility from the Oracle RAC database home, or use the **Convert to RAC** option on the single-instance database target of Oracle Enterprise Manager Cloud Control.

- Converting a single-instance database that uses a release of Oracle Database earlier than Oracle Database 12c to an Oracle RAC 12c database, running out of the same Oracle home and using the same data files as the single-instance database

In this scenario, use Oracle Universal Installer and Database Upgrade Assistant (DBUA) to update the single-instance database to Oracle Database 12c. Then use `rconfig` or the Oracle Enterprise Manager **Convert to RAC** option, as described in the preceding scenario.

- Converting a single-instance Oracle Database 12c to an Oracle RAC 12c database, running out of a *different* Oracle home and using the same data files as the single-instance database.

In this scenario, run the `rconfig` utility in the target database home, or use the **Convert to RAC** option in the single-instance database target of Oracle Enterprise Manager Cloud Control. Provide the file storage location when prompted.

Note:

If you specify Oracle home users for both the target database home and the source database home, then the Oracle home user for the target database home must be the same as the Oracle home user for the source database home.

- Converting a single-instance Oracle Database 12c to an Oracle RAC 12c database, running out of a *different* Oracle home, and where the host on which the single-instance database runs is *not* a node used by the Oracle RAC database.

In this scenario, create a clone image of the single-instance database, and move the clone image to a host that is one of the nodes used by the Oracle RAC database. Then use `rconfig` or the Oracle Enterprise Manager **Convert to RAC** option, as described in the preceding scenario.

Converting Databases to Oracle RAC Using Oracle Enterprise Manager

You can use Oracle Enterprise Manager Cloud Control to convert a single-instance database to an Oracle RAC database.

To use this feature, complete the following steps:

1. Log in to Oracle Enterprise Manager Cloud Control. From the Home page, click the **Targets** tab.
2. On the Targets page, click the **Databases** secondary tab, and click the link in the Names column of the database to convert to Oracle RAC.
3. On the Database home page, from the Availability menu, select **Convert to Cluster Database**.
4. Log in as the database user `SYS` with `SYSDBA` privileges to the database you want to convert, and click **Next**.
5. On the Convert to Cluster Database: Cluster Credentials page, provide a user name and password for the `oracle` user and password of the target database to convert. If the target database is using Oracle ASM, then also provide the `SYSASM` user and password, and then click **Next**.
6. On the Hosts page, select the host nodes in the cluster to be cluster members in the installed Oracle RAC database. When you have completed your selection, click **Next**.
7. On the Convert to Database: Options page, select whether you want to use the existing listener and port number, or specify a new listener and port number for the cluster. Also provide a prefix for cluster database instances on the cluster.

When you have finished entering information, click **Next**, or click **Help** if you need assistance in deciding how to enter information.

8. On the Convert to Cluster Database: Shared Storage page, either select the option to use your existing shared storage area, or select the option to have your database files copied to a new shared storage location. Also, decide if you want to use your existing fast recovery area or copy your recovery files to a new fast recovery area using files managed by Oracle Database.

If you use Oracle ASM, then Oracle recommends that you place the data files and the recovery files in separate failure groups. A failure group is defined by shared hardware, such as a controller shared between two disks, or two disks that are on the same spindle. If two disks share hardware that could fail, making both disks unavailable, then these disks are said to be in the same failure group. If you do not use Oracle ASM, then Oracle recommends that the data files and the recovery files are stored in separate locations, for example, separate Oracle ASM failure groups, so that a hardware failure does not affect availability.

When you have finished entering information, click **Next**, or click **Help** if you need assistance in deciding how to enter information.

9. On the Convert to Cluster Database: Review page, review the options you have selected. Click **Submit Job** to proceed with the conversion. To change any options you have selected, click **Back**. To cancel the conversion, click **Cancel**.
10. On the Confirmation page, click **View Job** to check the status of the conversion.

Converting Databases to Oracle RAC Using rconfig

You can use the command-line utility `rconfig` to convert a single-instance database to an Oracle RAC database, or to convert it to an Oracle RAC One Node database, depending on the values you provide in the `ConvertToRAC.xml` file. To use this feature, complete the following steps:

1. As the `oracle` user, navigate to the directory `$ORACLE_HOME/assistants/rconfig/sampleXMLs`, and open the file `ConvertToRAC.xml` using a text editor, such as `vi`.
2. Review the `ConvertToRAC.xml` file, and modify the parameters as required for your system. The XML sample file contains comments that provide instructions for how to configure the file.

▲ Caution:

Set the convert option `Convert verify="ONLY"` to perform a test conversion to ensure that a conversion can be completed successfully.

When you have finished modifying parameters, save the file with a name of the format `file_name.xml`. Make a note of the name you select.

3. Navigate to the directory `$ORACLE_HOME/bin`, and use the following command to run the command `rconfig`, where `input.xml` is the name of the XML input file you configured in Step 2:

```
rconfig input.xml
```

For example, if you create an input XML file called `convert.xml`, then you would use the following command

```
$ ./rconfig convert.xml
```

✎ Note:

The `Convert verify` option in the `ConvertToRAC.xml` file has three options:

- `Convert verify="YES"`: `rconfig` performs checks to ensure that the prerequisites for single-instance to Oracle RAC conversion have been met before it starts conversion
- `Convert verify="NO"`: `rconfig` does not perform prerequisite checks, and starts conversion
- `Convert verify="ONLY"`: `rconfig` only performs prerequisite checks; it does not start conversion after completing prerequisite checks

If performing the conversion fails, then use the following procedure to recover and reattempt the conversion.:

1. Attempt to delete the database using the DBCA delete database option.
2. Restore the source database.

3. Review the conversion log, and fix any problems reported by `rconfig` that may have caused the conversion failure. The `rconfig` log files are under the `rconfig` directory in `$ORACLE_BASE/cfgtoollogs`.
4. Reattempt the conversion.

Example of rconfig XML Input Files for ConvertToRAC

The following are two examples of an XML `ConvertToRAC` input file for the `rconfig` utility. [Example 15-1](#) is an XML input file to convert a single-instance database with Oracle ASM to a policy-managed Oracle RAC database (using server pools) on Oracle ASM storage. [Example 15-2](#) is an XML input file to convert a single-instance database with Oracle ASM to an administrator-managed Oracle RAC database.

Example 15-1 Example rconfig ConvertToRAC XML File for Policy-Managed Databases

```
<?xml version="1.0" encoding="UTF-8"?>
<n:RConfig xmlns:n="http://www.example.com/rconfig"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://www.example.com/rconfig">
  <n:ConvertToRAC>
    <!-- Verify does a precheck to ensure all pre-requisites are met, before the
         conversion is attempted. Allowable values are: YES|NO|ONLY -->
    <n:Convert verify="YES">
    <!--Specify current OracleHome of non-rac database for SourceDBHome -->
      <n:SourceDBHome>/oracle/product/12.1.0/db_1</n:SourceDBHome>
    <!--Specify OracleHome where the rac database should be configured. It can be same
         as SourceDBHome -->
      <n:TargetDBHome>/oracle/product/12.1.0/db_1</n:TargetDBHome>
    <!--Specify SID of non-rac database and credential. User with sysdba role is
         required to perform conversion -->
      <n:SourceDBInfo SID="sales">
        <n:Credentials>
          <n:User>sys</n:User>
          <n>Password>oracle</n>Password>
          <n:Role>sysdba</n:Role>
        </n:Credentials>
      </n:SourceDBInfo>
    <!--Specify the list of existing or new server pools which are used by the
         Policy Managed Cluster Database. -->
      <n:ServerPoolList>
        <n:ExistingServerPool name="custom"/>
        <n:NewServerPool name="newpool" cardinality="2"/>
      </n:ServerPoolList>
    <!--Specify RacOneNode along with servicename to convert database to RACOne
         Node -->
      <!--n:RacOneNode servicename="salesraclservice"/-->
    <!--InstancePrefix is not required for Policy Managed database. If specified, it
         will be ignored. Instance names are generated automatically based on db_unique_
         name for Policy Managed dababase.-->
    <!-- Listener details are no longer needed starting 11.2. Database is registered
         with default listener and SCAN listener running from Oracle Grid Infrastructure
         home. -->
    <!--Specify the type of storage to be used by rac database. Allowable values are
         CFS|ASM. The non-rac database should have same storage type. ASM credentials are
         no needed for conversion. -->
      <n:SharedStorage type="ASM">
    <!--Specify Database Area Location to be configured for rac database.If this field
         is left empty, current storage will be used for rac database. For CFS, this field
```

```

will have directory path. -->
    <n:TargetDatabaseArea>+ASMDG</n:TargetDatabaseArea>
<!--Specify Fast Recovery Area to be configured for rac database. If this field is
left empty, current recovery area of non-rac database will be configured for rac
database. If current database is not using recovery Area, the resulting rac
database will not have a recovery area. -->
    <n:TargetFlashRecoveryArea>+ASMDG</n:TargetFlashRecoveryArea>
    </n:SharedStorage>
</n:Convert>
</n:ConvertToRAC>
</n:RConfig>

```

Example 15-2 Example rconfig ConvertToRAC XML File for Administrator-Managed Databases

```

<?xml version="1.0" encoding="UTF-8"?>
<n:RConfig xmlns:n="http://www.example.com/rconfig"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.example.com/rconfig rconfig.xsd">
    <n:ConvertToRAC>
<!-- Verify does a precheck to ensure all pre-requisites are met, before the
conversion is attempted. Allowable values are: YES|NO|ONLY -->
        <n:Convert verify="YES">
<!--Specify current OracleHome of non-rac database for SourceDBHome -->
            <n:SourceDBHome>/oracle/product/12.1.0/db_1</n:SourceDBHome>
<!--Specify OracleHome where the rac database should be configured. It can be same
as SourceDBHome -->
            <n:TargetDBHome>/oracle/product/12.1.0/db_1</n:TargetDBHome>
<!--Specify SID of non-rac database and credential. User with sysdba role is
required to perform conversion -->
            <n:SourceDBInfo SID="sales">
                <n:Credentials>
                    <n:User>sys</n:User>
                    <n>Password>oracle</n>Password>
                    <n:Role>sysdba</n:Role>
                </n:Credentials>
            </n:SourceDBInfo>
<!--Specify the list of nodes that should have rac instances running for the Admin
Managed Cluster Database. LocalNode should be the first node in this nodelist.
-->
                <n:NodeList>
                    <n:Node name="node1"/>
                    <n:Node name="node2"/>
                </n:NodeList>
<!--Specify RacOneNode along with servicename to convert database to RACOne
Node -->
                <n:RacOneNode servicename="salesraclservice"/-->
<!--Instance Prefix tag is optional starting with 11.2. If left empty, it is
derived from db_unique_name.-->
                <n:InstancePrefix>sales</n:InstancePrefix>
<!-- Listener details are no longer needed starting 11.2. Database is registered
with default listener and SCAN listener running from Oracle Grid Infrastructure
home. -->
                <n:Storage type="ASM">
<!--Specify the type of storage to be used by rac database. Allowable values are
CFS|ASM. The non-rac database should have same storage type. ASM credentials
are not needed for conversion. -->
                    <n:SharedStorage type="ASM">
<!--Specify Database Area Location to be configured for rac database.If this field
is left empty, current storage will be used for rac database. For CFS, this
field will have directory path. -->
                        <n:TargetDatabaseArea>+ASMDG</n:TargetDatabaseArea>

```



```
<!--Specify Fast Recovery Area to be configured for rac database. If this field is
left empty, current recovery area of non-rac database will be configured for rac
database. If current database is not using recovery Area, the resulting rac
database will not have a recovery area. -->
    <n:TargetFlashRecoveryArea>+ASMDG</n:TargetFlashRecoveryArea>
    </n:SharedStorage>
    </n:Convert>
    </n:ConvertToRAC>
</n:RConfig>
```

Postconversion Steps

After completing the conversion, note the following recommendations for Oracle RAC environments, as described in the following:

- Follow the recommendations for using load balancing and transparent application failover, as previously described
- Use locally managed tablespaces instead of dictionary managed tablespaces to reduce contention and manage sequences in Oracle RAC as described in *Oracle Database Administrator's Guide*
- Follow the guidelines for using automatic segment space management as described in *Oracle Database Administrator's Guide*

The buffer cache and shared pool capacity requirements in Oracle RAC are slightly greater than the requirements for single-instance Oracle databases. Therefore, you should increase the size of the buffer cache by about 10%, and the size of the shared pool by about 15%.

A

Server Control Utility Reference

Use the Server Control Utility (SRVCTL) to manage Oracle Real Application Clusters (Oracle RAC) configuration information.

Note:

SRVCTL commands specific to Oracle Grid Infrastructure administration operations are documented in *Oracle Clusterware Administration and Deployment Guide*

This appendix includes the following topics:

- [SRVCTL Usage Information](#)
- [Specifying Command Parameters as Keywords Instead of Single Letters](#)
- [Character Set and Case Sensitivity of SRVCTL Object Values](#)
- [Summary of Tasks for Which SRVCTL Is Used](#)
- [Using SRVCTL Help](#)
- [SRVCTL Privileges and Security](#)
- [Additional SRVCTL Topics](#)
- [Deprecated SRVCTL Subprograms or Commands](#)
- [SRVCTL Command Reference](#)

SRVCTL Usage Information

SRVCTL is installed on each node in a cluster by default. To use SRVCTL, log in to the operating system of a node and enter the SRVCTL command and its parameters in case-sensitive syntax.

- Only use the version of SRVCTL that is provided with Oracle Database 12c on Oracle RAC databases that are created for or upgraded to Oracle Database 12c. The version of SRVCTL must be the same as the version of the object (listeners, Oracle ASM instances, Oracle RAC databases and their instances, and services) being managed. To ensure the versions are the same release, run SRVCTL commands from the Oracle home of the database or object you are managing.
- SRVCTL does not support concurrent executions of commands on the same object. Therefore, run only one SRVCTL command at a time for each database, service, or other object.
- When specifying a comma-delimited list as part of a SRVCTL command, there should not be any spaces between the items in the list. For example:

```
srvctl add database -serverpool "serverpool1,serverpool3"
```

When you specify a comma-delimited list in a Windows environment, you must enclose the list in double quotation marks (""). You can enclose a comma-delimited list in double quotation marks in a Linux or UNIX environment but they will be ignored.

- If you are entering a SRVCTL command, and you want to continue the input on a new line, then you can use the operating system continuation character. In Linux, this is the backslash (\) symbol.
- A SRVCTL command that produces no output is a successful command. Not all SRVCTL commands return a message when it completes, successfully. However, if a SRVCTL command fails, then it always returns an error message.
- You can use the `-eval` parameter with several SRVCTL commands. This parameter, when you use it, enables you to simulate running a command without making any changes to the system. SRVCTL returns output that informs you what will happen if you run a particular command. For example, to know what might happen if you relocate a server:

```
$ srvctl relocate server -servers "rac1" -eval -serverpool pool2
```

```
Database db1
  will stop on node rac1
  will start on node rac7
  Service mySrv1
    will stop on node rac1, it will not run on any node
  Service myServ2
    will stop on node rac1
    will start on node rac6
Server rac1
  will be moved from pool myPoolX to pool pool2
```

The `-eval` parameter is available with the following commands:

- `srvctl add database`
- `srvctl add service`
- `srvctl add srvpool`
- `srvctl modify database`
- `srvctl modify service`
- `srvctl modify srvpool`
- `srvctl relocate server`
- `srvctl relocate service`
- `srvctl remove srvpool`
- `srvctl start database`
- `srvctl start service`
- `srvctl stop database`
- `srvctl stop service`

Specifying Command Parameters as Keywords Instead of Single Letters

Before Oracle Database 12c, the SRVCTL command-line interface used single letter parameters. However, this imposes a limit on the number of unique parameters available for use with SRVCTL commands. SRVCTL command parameters introduced in Oracle Database 12c are full words instead of single letters, such as `-multicastport` and `-subdomain`.

To support backward compatibility, you can use a mix of single-letter parameters and new keyword parameters. New parameters introduced with keywords can be used with single letter parameters.

 **Note:**

Starting with Oracle Database 12c, the single letter parameters are deprecated in favor of the keyword parameters to avoid using the same letter to implement different functionality depending on the command.

You can obtain the single-letter equivalents, where applicable, by adding the `-compatible` parameter after the `-help` parameter.

Character Set and Case Sensitivity of SRVCTL Object Values

SRVCTL interacts with many different types of objects. The character set and name length limitations, and whether the object name is case sensitive, can vary between object types.

Table A-1 String Restrictions for SRVCTL Object Names

Object Type	Character Set Limitations	Case Sensitive?	Maximum Length
db_domain	Alpha-numeric characters, underscore (<code>_</code>), and number sign (<code>#</code>)		128 characters
db_unique_name	Alpha-numeric characters, underscore (<code>_</code>), number sign (<code>#</code>), and dollar sign (<code>\$</code>); the first 8 characters must be unique because those characters are used to form instance names for policy-managed databases	No	30 characters but the first 8 characters must be unique relative to any other database in the same cluster

Table A-1 (Cont.) String Restrictions for SRVCTL Object Names

Object Type	Character Set Limitations	Case Sensitive?	Maximum Length
diskgroup_name	Naming disk groups have the same limitations as naming other database objects. See Also: <i>Oracle Database SQL Language Reference</i> for more information about database object naming rules	No (all names are converted to uppercase)	
instance_name	Alpha-numeric characters	Depends on the platform	15 characters
listener_name			
node_name		No	
scan_name	The first character must be an alphabetic character	No	
server_pool	Alpha-numeric characters, underscore (_), number sign (#), period (.), and dollar sign (\$); the name cannot begin with a period, contain single quotation marks (' '), nor can the name be "Generic" or "Free" because those two names are reserved for the built-in server pools		250 characters
service_name			250 characters
volume_name	Alphanumeric characters; dashes (-) are not allowed and the first character must be an alphabetic character.	No	11 characters

Summary of Tasks for Which SRVCTL Is Used

Use SRVCTL to manage databases, instances, cluster databases, cluster database instances, Oracle ASM instances and disk groups, services, listeners, or other clusterware resources.

- Cluster Database Configuration Tasks

Tasks	Commands
Add, modify, and delete cluster database configuration information	<pre> srvctl add database srvctl modify database srvctl remove database </pre>

Tasks	Commands
Add an instance to or delete an instance from the configuration of a cluster database	<pre>srvctl add instance srvctl remove instance</pre>
Add a service to or delete a service from the configuration of a cluster database	<pre>srvctl add service srvctl remove service</pre>
Move instances and services in a cluster database configuration and modify service configurations	<pre>srvctl relocate database srvctl relocate service srvctl modify instance srvctl modify service</pre>
Set and unset the environment for an instance or service in a cluster database configuration	<pre>srvctl modify instance srvctl modify service</pre>
Set and unset the environment for an entire cluster database in a cluster database configuration	<pre>srvctl setenv database srvctl unsetenv database</pre>

- General Cluster Database Administration Tasks

Tasks	Commands
Start and stop cluster databases	<pre>srvctl start database srvctl stop database</pre>
Start and stop cluster database instances	<pre>srvctl start instance srvctl stop instance</pre>
Start, stop, and relocate cluster database services	<pre>srvctl start service srvctl stop service srvctl relocate service</pre>
Obtain statuses of cluster databases, cluster database instances, or cluster database services	<pre>srvctl status database srvctl status instance srvctl status service</pre>

- Node-Level Tasks

Tasks	Commands
Administering VIPs	<pre>srvctl add vip srvctl config vip srvctl disable vip srvctl enable vip srvctl getenv vip srvctl modify vip srvctl relocate vip srvctl remove vip srvctl setenv vip srvctl start vip srvctl status vip srvctl stop vip srvctl unsetenv vip</pre>
Administering node applications	<pre>srvctl add nodeapps srvctl disable nodeapps srvctl enable nodeapps srvctl getenv nodeapps srvctl modify nodeapps srvctl remove nodeapps srvctl setenv nodeapps srvctl unsetenv nodeapps</pre>

Using SRVCTL Help

This section includes information about using context sensitive help with SRVCTL.

To see help for all SRVCTL commands, from the command line enter:

```
srvctl -help
```

To see the command syntax and a list of parameters for each SRVCTL command, from the command line enter:

```
srvctl command (or verb) object (or noun) -help
```

When you request online help for a command using `-help`, SRVCTL prints the full words for each parameter. You can obtain the single-letter equivalents, where applicable, by adding the `-compatible` parameter after the `-help` parameter. For example:

```
$ srvctl config database -help -compatible
```

The preceding command prints usage information for the `srvctl config database` command, listing all parameters as full words followed by their single-letter equivalents in parentheses, where applicable.

To see the SRVCTL version number enter:

```
$ srvctl -version
```

SRVCTL Privileges and Security

To use SRVCTL to change your Oracle RAC database configuration, log in to the operating system as the software owner of the home that you want to manage.

For example, if different users installed Oracle Database and the Oracle Grid Infrastructure, then log in as the database software owner (for example, `ora_db`) to manage databases and log in as the Oracle Grid Infrastructure software owner (for example, `ora_asm`) to manage the Oracle ASM instances.

Users who are members of the OSDBA operating system group can start and stop the database. To stop and start an Oracle ASM instance, you must be a member of the OSASM operating system group.

To create or register objects such as listeners, Oracle Notification Services, and services, you must be logged in to the operating system as the software owner of the Oracle home. The objects you create or register for that Oracle home will run under the user account of the owner of the Oracle home. Databases run as the database installation owner of the home from which they run.

To perform `srvctl add` operations on any object, you must be logged in as the Oracle account owner of the home on which the object runs.

For some SRVCTL commands, 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 to run them. In this appendix, those commands are preceded by the root prompt (`#`) in the command examples.

Additional SRVCTL Topics

- Use SRVCTL to manage Oracle-supplied resources such as listener, instances, disk groups, and networks, and CRSCTL for managing Oracle Clusterware and its resources.

Note:

Oracle strongly discourages directly manipulating Oracle-supplied resources (resources whose names begin with `ora`) using CRSCTL. This could adversely impact the cluster configuration.

- Although you may be able to cancel running SRVCTL commands by pressing the Control-C keys, you may corrupt your configuration data by doing this.

You are strongly advised *not* to attempt to terminate SRVCTL in this manner.

Deprecated SRVCTL Subprograms or Commands

A number of SRVCTL commands and parameters have been deprecated in this release.

Single Character Parameters for all SRVCTL Commands

Single-character parameters have been deprecated in Oracle Database 12c.

Use the full keyword for each parameter instead. To support older tools and scripts that still use single-character parameters, the current version of SRVCTL supports both single-character parameters and full keyword parameters.

The command reference in this appendix shows the keywords for each SRVCTL command. [Table A-2](#) lists the deprecated single-character parameters.

Table A-2 Depreciated Single-Character Parameters for SRVCTL Commands

Single Letter	Long Form	Values	Description	Related Commands
A	address	{VIP_name IP}/ netmask/ [if1 if2...]]	VIP address specification for node applications	Node applications, VIP, network, Listener, SCAN VIP, and SCAN listener commands
a	all		All resources of that kind	srvctl config database Common
a	diskgroup	diskgroup_list	Comma-delimited list of Oracle ASM disk groups	Database, instance, Oracle ASM, disk group, and file system commands
a	detail		Print detailed configuration information	Common
a	available	available_list	A comma-delimited list of available instances	Service and server pool commands
a	abort		Abort failed online relocation	Relocate database
a	viponly		Display VIP configuration	Node applications, VIP, network, listener, SCAN VIP, and SCAN listener commands
B	rlbgoal	{NONE SERVICE_TIME THROUGHPUT}	The runtime load balancing goal of a service	Service and server pool commands
c	currentnode	current_node	Node name from which to relocate the service	Service and server pool commands
c	cardinality	{UNIFORM SINGLETON}	Whether the service should run on every active server in the server pool (UNIFORM) or just one server (SINGLETON)	Service and server pool commands

Table A-2 (Cont.) Deprecated Single-Character Parameters for SRVCTL Commands

Single Letter	Long Form	Values	Description	Related Commands
c	dbtype	<i>type</i>	Type of database: Oracle RAC One Node, Oracle RAC, or single instance	Database, instance, Oracle ASM, disk group, and file system commands
d	db or database	<i>db_unique_name</i>	Database unique name	Common
d	device	<i>volume_device</i>	Volume device path	Database, instance, Oracle ASM, disk group, and file system commands
d	domain		Display subdomain served by GNS	OC4J, home, CVU, and GNS commands
e	emport	<i>em_port_number</i>	Local listen port for Oracle Enterprise Manager	Node applications, VIP, network, listener, SCAN VIP, and SCAN listener commands
e	failovertyp e	{NONE SESSION BASIC TRANSACTION}	The failover type for a service	Service and server pool commands
e	server	<i>server_list</i>	Candidate server list for Oracle RAC One Node database	Database, instance, Oracle ASM, disk group, and file system commands
f	force		Force remove	Common
g	diskgroup	<i>diskgroup_name</i>	Disk group name	File system, Diskgroup commands
g	gsdonly		Display GSD configuration	Node applications, VIP, network, listener, SCAN VIP, and SCAN listener commands
g	serverpool	<i>server_pool_name</i> <i>server_pool_list</i>	A server pool name Comma-delimited list of database server pool names	Service and server pool commands Database, instance, Oracle ASM, disk group, and file system commands
h	help			Common
i	importance	<i>number</i>	A number that represents the importance of the server pool	Service and server pool commands
i	instance	<i>instance_name</i> <i>instance_list</i>	Instance name prefix for administrator-managed Oracle RAC One Node database A comma-delimited list of instance names	Database, instance, Oracle ASM, disk group, and file system commands

Table A-2 (Cont.) Deprecated Single-Character Parameters for SRVCTL Commands

Single Letter	Long Form	Values	Description	Related Commands
I	ip	<i>ip_address</i>	VIP address on which GNS is to listen	OC4J, home, CVU, and GNS commands
i	oldinst	<i>instance_name</i>	The old instance name	Service and server pool commands
i	scannumber	<i>scan_ordinal_number</i>	Ordinal number of the IP address for the SCAN	Node applications, VIP, network, listener, SCAN VIP, and SCAN listener commands
i	vip	<i>vip_name</i> or " <i>vip_name_list</i> "	VIP names	Node applications, GNS, VIP, network, listener, SCAN VIP, and SCAN listener commands
j	acfspath	<i>acfs_path_list</i>	Comma-delimited list of Oracle ACFS paths where the dependency on the database will be set	Database, instance, Oracle ASM, disk group, and file system commands
j	clbgoal	{SHORT LONG}	The connection load balancing goal for a service	Service and server pool commands
k	netnum	<i>network_number</i>	The network number	Service and server pool commands Node applications, VIP, network, listener, SCAN VIP, and SCAN listener commands OC4J, home, CVU, and GNS commands
l	list		List all records in GNS	OC4J, home, CVU, and GNS commands
l	listener	<i>listener_name</i>	The name of a listener	ASM commands
l	loglevel	<i>log_level</i>	Specify the level (0-6) of logging that GNS should run with	OC4J, home, CVU, and GNS commands
l	min	<i>number</i>	The minimum size of the server pool	Service and server pool commands
l	onslocalport	<i>port_number</i>	Oracle Notification Service listening port for local client connections	Node applications, VIP, network, listener, SCAN VIP, and SCAN listener commands

Table A-2 (Cont.) Deprecated Single-Character Parameters for SRVCTL Commands

Single Letter	Long Form	Values	Description	Related Commands
l	role	<i>service_role</i>	Comma-delimited list of server roles within double quotation marks (" "), where each role is one of PRIMARY, PHYSICAL_STANDBY, LOGICAL_STANDBY, or SNAPSHOT_STANDBY	Service and server pool commands
m	domain	<i>domain_name</i>	The domain for the database	Database, instance, Oracle ASM, disk group, and file system commands
m	failovermethod	{NONE BASIC}	The failover method of a service	Service and server pool commands
m	multicastport		The port on which the GNS daemon is listening for multicast requests	OC4J, home, CVU, and GNS commands
m	path	<i>mountpoint_path</i>	Mountpoint path	Database, instance, Oracle ASM, disk group, and file system commands
n	name		Advertise a name through GNS using the given address	OC4J, home, CVU, and GNS commands
n	node	<i>node_name</i>	The name of a specific node	Common
n	nodes	<i>node_list</i>	A comma-delimited list of node names	File system commands
n	dbname	<i>database_name</i>	The database name (DB_NAME), if different from the unique name specified by the -db parameter	Database, instance, Oracle ASM, disk group, and file system commands
n	scanname	<i>scan_name</i>	Fully-qualified SCAN name (includes the domain)	Node applications, VIP, network, listener, SCAN VIP, and SCAN listener commands
n	servers	<i>server_list</i>	A comma-delimited list of candidate server names	Service and server pool commands
n	targetnode	<i>node_name</i>	Node name to which to relocate the service	Service and server pool commands
o	oraclehome	<i>oracle_home</i>	<i>\$ORACLE_HOME</i> path	Database commands

Table A-2 (Cont.) Deprecated Single-Character Parameters for SRVCTL Commands

Single Letter	Long Form	Values	Description	Related Commands
p	endpoints	[TCP:] <i>port_number</i> [/IPC: <i>key</i> [/NMP: <i>pipe_name</i> [/TCPS: <i>s_port</i> [/SDP: <i>port</i>]	SCAN listener endpoints	Node applications, VIP, network, listener, SCAN VIP, and SCAN listener commands
p	port		The port which the GNS daemon uses to communicate with the DNS server	OC4J, home, CVU, and GNS commands
p	rmiport	<i>port_number</i>	OC4J RMI port number	OC4J, home, CVU, and GNS commands
P	tafpolicy	{NONE BASIC}	TAF policy specification	Service and server pool commands
p	spfile	<i>spfile_location</i>	Server parameter file path	Database, instance, Oracle ASM, disk group, and file system commands
q	notification	{TRUE FALSE}	Whether FAN is enabled for OCI connections	Service commands
q	query		Query GNS for the records belonging to a name	OC4J, home, CVU, and GNS commands
r	preferred	<i>preferred_list</i>	A comma-delimited list of preferred instances	Service and server pool commands
r	onsremotepo rt	<i>port_number</i>	Oracle Notification Service listening port for connections from remote hosts	Node applications, VIP, network, listener, SCAN VIP, and SCAN listener commands
r	relocate		Relocate the VIP	Node applications, VIP, network, listener, SCAN VIP, and SCAN listener commands
r	revert		Remove target node of failed online relocation request from the candidate server list of administrator-managed Oracle RAC One Node database	Relocate database
r	role	<i>role_type</i>	Role of the standby database: PRIMARY, PHYSICAL_STANDBY, LOGICAL_STANDBY, or SNAPSHOT_STANDBY	Database, instance, Oracle ASM, disk group, and file system commands

Table A-2 (Cont.) Deprecated Single-Character Parameters for SRVCTL Commands

Single Letter	Long Form	Values	Description	Related Commands
s	ononly		Display Oracle Notification Service daemon configuration	Node applications, VIP, network, listener, SCAN VIP, and SCAN listener commands
s	skip		Skip checking the ports	Listener, SCAN, and SCAN listener.
s	statfile	<i>file_name</i>	The file path of the <i>state_file</i> created by a previously executed <code>srvctl stop home</code> command	OC4J, home, CVU, and GNS commands
s	status		Display the status of GNS	OC4J, home, CVU, and GNS commands
S	subnet	<i>subnet/net_mask</i> [[<i>if1</i> [[<i>if2</i> ...]]]	Network address specification for a network	Node applications, VIP, network, listener, SCAN VIP, and SCAN listener commands
s	service	<i>service_name</i> <i>service_name_list</i>	The name of a service A comma-delimited list of service names	Service and server pool commands
s	startoption	<i>start_options</i>	Startup options for the database (mount, open, read only)	Database, instance, Oracle ASM, disk group, and file system commands
t	checkinterval	<i>time_interval</i>	Interval in minutes between checks	OC4J, home, CVU, and GNS commands
t	edition	<i>edition_name</i>	The initial session edition of a service	Service and server pool commands
t	envs	" <i>name_list</i> "	A list of environment variables	Common
t	namevals	" <i>name= value,...</i> "	Names and values of environment variables	Common
T	nameval	" <i>name=value</i> "	Name and value of a single environment variable	Common
t	update	<i>instance_name</i>	The new instance name	Service and server pool commands
t	remoteservers	<i>host_name</i> [: <i>port_number</i>] [, <i>host_name</i> [: <i>port_number</i>]...]	List of remote host name and port number pairs for Oracle Notification Service daemons outside this cluster	Node applications, VIP, network, listener, SCAN VIP, and SCAN listener commands

Table A-2 (Cont.) Deprecated Single-Character Parameters for SRVCTL Commands

Single Letter	Long Form	Values	Description	Related Commands
t	stopoption	<i>stop_options</i>	Stop options for the database (NORMAL, TRANSACTIONAL, IMMEDIATE, or ABORT)	Database, instance, Oracle ASM, disk group, and file system commands
t	toversion	<i>target_version</i>	Version to which you are downgrading	Database, instance, Oracle ASM, disk group, and file system commands
u	max	<i>number</i>	Maximum size of the server pool	Service and server pool commands
u	nettype	<i>network_type</i>	The network server type, which can be STATIC, DHCP, or MIXED	Node applications, VIP, network, listener, SCAN VIP, and SCAN listener commands
u	newinst		Add a new instance to the service configuration	Service commands
u	update		Update SCAN listeners to match the number of SCAN VIPs	Node applications, VIP, network, listener, SCAN VIP, and SCAN listener commands
u	user	<i>oracle_user</i>	Oracle user or other authorized user to mount and unmount file systems	Database, instance, Oracle ASM, disk group, and file system commands
v	verbose		Verbose output	Common
v	volume	<i>volume_name</i>	Name of a volume	Database, instance, Oracle ASM, disk group, and file system commands
V	versions			Common
w	failoverdelay	<i>number</i>	Failover delay	Service and server pool commands
w	nettype	<i>network_type</i>	The network server type, which can be STATIC, DHCP, or MIXED	Node applications, VIP, network, listener, SCAN VIP, and SCAN listener commands
w	timeout	<i>timeout</i>	Online relocation timeout in minutes	Database, instance, Oracle ASM, disk group, and file system commands
x	dtp	{TRUE FALSE}	Whether to enable distributed transaction processing	Service and server pool commands
x	node	<i>node_name</i>	Node name (use this parameter only with noncluster databases)	Common

Table A-2 (Cont.) Deprecated Single-Character Parameters for SRVCTL Commands

Single Letter	Long Form	Values	Description	Related Commands
y	noprompt		Suppress the confirmation prompt	Common
y	policy	{AUTOMATIC MANUAL}	Management policy for the resource	Database, instance, Oracle ASM, disk group, file system, service and server pool commands
z	failoverretry	number	Number of failover retries	Service and server pool commands
z	rmdepondisk		To remove a database's dependency upon disk groups	Database, instance, Oracle ASM, disk group, and file system commands

Miscellaneous SRVCTL Commands and Parameters

The following command parameters have been deprecated in this release:

Table A-3 Deprecated Commands and Parameters for SRVCTL

Command	Deprecated Parameters
srvctl modify asm	-node <i>node_name</i>
srvctl modify instance	-z Instead, use the -node option with the value set to ""
srvctl modify gns	[-ip <i>ip_address</i>] [-advertise <i>host_name</i> -address <i>address</i>] [-delete <i>host_name</i> -address <i>address</i>] [-createalias <i>name</i> -alias <i>alias</i>] [-deletealias <i>alias</i>] Use the <code>srvctl update gns</code> command instead.
srvctl * oc4j	The <code>oc4j</code> noun has been deprecated and replaced with <code>qosmserver</code> . SRVCTL still accepts the <code>oc4j</code> noun until it is desupported.
srvctl add service	The <code>PRECONNECT</code> option with the <code>-tafpolicy</code> parameter is deprecated.
srvctl modify service	The <code>-failovermethod</code> {NONE BASIC} is deprecated. The <code>PRECONNECT</code> option with the <code>-tafpolicy</code> parameter is deprecated.

SRVCTL Command Reference

A comprehensive list of SRVCTL commands to use in Oracle RAC environments.

SRVCTL commands, object names, and parameters are case sensitive. Database, instance, listener, and service names are case insensitive and case preserving. You

cannot create listener names that differ only in case, such as LISTENER and listener. SRVCTL uses the following command syntax:

```
srvctl command object [parameters]
```

In SRVCTL syntax:

- *command* is a verb such as `start`, `stop`, or `remove`
- *object* (also known as a *noun*) is the target or object on which SRVCTL performs the command, such as database or instance. You can also use object abbreviations.
- *parameters* extend the use of a preceding command combination to include additional parameters for the command. For example, the `-instances` parameter indicates that a comma-delimited list of preferred instance names follows; the `-instance` parameter only permits one value and not a list of names. Do not use spaces between the items in a comma-delimited list.

 **Note:**

If specifying a comma-delimited list in Windows, then you must enclose the list within double quotation marks ("").

Table A-4 lists the keywords that can be used for the *object* portion of SRVCTL commands. You can use either the full name or the abbreviation for each object keyword. The **Purpose** column describes the object and the actions that can be performed on that object.

Table A-4 Object Keywords and Abbreviations

Object	Keyword	Purpose
Database	database	To add, modify, manage environment variables for, list the configuration of, enable, disable, start, stop, and obtain the status of databases, and also to upgrade, downgrade, and remove database configuration information about databases.
Instance	instance	To add, modify, enable, disable, start, stop, obtain the status of, and remove database instances.
	inst	
Listener	listener	To add, modify, manage environment variables for, list the configuration of, enable, disable, start, stop, obtain the status of, and remove listeners
	lsnr	
Network	network	To add, modify, list the configuration of, and remove a non-default Network Note: The <code>node applications</code> object, and the <code>config</code> and <code>modify</code> commands also manage the default network.

Table A-4 (Cont.) Object Keywords and Abbreviations

Object	Keyword	Purpose
Node applications	nodeapps	To add, modify, manage environment variables for, list the configuration of, enable, disable, start, stop, obtain the status of, and remove node applications
Oracle Notification Service	ons	To add, configure, enable, start, obtain the status of, stop, disable, and remove Oracle Notification Service instances only for Oracle Restart
Single client access name (SCAN)	scan	To add, list the configuration of, modify, enable, disable, start, stop, relocate, obtain the status of, and remove SCAN VIPs
SCAN listener	scan_listener	To add, list the configuration of, modify, enable, disable, start, stop, relocate, obtain the status of, and remove SCAN listeners
Service	service	To add, modify, list the configuration of, enable, disable, start, stop, obtain the status of, relocate, and remove services
Virtual IP	vip	To add, manage environment variables for, list the configuration of, enable, disable, start, stop, obtain the status of, and remove a VIP

 **Note:**

SRVCTL commands specific to Oracle Grid Infrastructure administration operations are documented in *Oracle Clusterware Administration and Deployment Guide*

srvctl add database

Adds a database configuration to Oracle Clusterware.

Syntax

```

srvctl add database -db db_unique_name [-eval]
    -oraclehome oracle_home [-node node_name] [-domain domain_name]
    [-spfile spfile] [-pwfile password_file_path]
    [-dbtype {RACONENODE | RAC | SINGLE}] [-server "server_list"]
    [-instance instance_name] [-timeout timeout]
    [-role {PRIMARY | PHYSICAL_STANDBY | LOGICAL_STANDBY | SNAPSHOT_STANDBY}]
    [-startoption start_options] [-stopoption stop_options] [-dbname db_name]
    [-acfspace "acfs_path_list"] [-policy {AUTOMATIC | MANUAL | NORESTART}]
    [-serverpool "server_pool_list" [-pqpools "pq_pool_list"]]
    [-diskgroup "disk_group_list" [-css_critical {yes | no}] [-cpucount
cpu_count]
    [-memorytarget memory_target] [-maxmemory max_memory] [-cpucap cpu_cap] [-
defaultnetnum network_number] [-verbose]

```

Parameters

Table A-5 srvctl add database Command Parameters

Parameter	Description
-db <i>db_unique_name</i>	Specify the unique name of the database.
-eval	Use this parameter to hypothetically evaluate the impact of the command on the system. Note: You can only use this parameter with a policy-managed database.
-oraclehome <i>oracle_home</i>	Specify the path for the Oracle database home directory.
-node <i>node_name</i>	Specify the node name on which you want to register a noncluster, or single instance, Oracle database. Note: This parameter can be used only with Oracle Clusterware and can be used with the <code>-serverpool</code> parameter to create a policy-managed, noncluster database.
-domain <i>db_domain</i>	Specify the domain for the database. Note: You must use this parameter if you set the <code>DB_DOMAIN</code> initialization parameter for the database.
-spfile <i>spfile</i>	Specify the path name of the database server parameter file.
-pwfile <i>password_file_path</i>	Enter the full path to the location of the password file.
-dbtype {RACONENODE RAC SINGLE}	Specify the type of database you are adding: Oracle RAC One Node, Oracle RAC, or single instance. The default is <code>RAC</code> unless you specify the <code>-node node_name</code> parameter, and the <code>-type</code> parameter defaults to <code>SINGLE</code> .
-server <i>server_list</i>	List candidate servers for Oracle RAC One Node databases. Note: You can use this parameter only with administrator-managed Oracle RAC One Node databases. If your Oracle RAC One Node database is policy managed, then you <i>cannot</i> use this parameter.
-instance <i>instance_name</i>	Specify the instance name prefix for Oracle RAC One Node databases. The default value for this parameter is the first 12 characters of the global unique name of the database. Note: You can use this parameter only with administrator-managed Oracle RAC One Node databases. If your Oracle RAC One Node database is policy managed, then you <i>cannot</i> use this parameter.
-timeout <i>timeout</i>	Specify the online database relocation timeout, in minutes, for Oracle RAC One Node databases. The default is 30.
-role {PRIMARY PHYSICAL_STANDBY LOGICAL_STANDBY SNAPSHOT_STANDBY}	Specify the role of the database in an Oracle Data Guard configuration. The default is <code>PRIMARY</code> .

Table A-5 (Cont.) `srvctl add database` Command Parameters

Parameter	Description
<code>-startoption</code> <code>start_options</code>	Startup options for the database, such as <code>OPEN</code> , <code>MOUNT</code> , and <code>NOMOUNT</code> . The default value is <code>OPEN</code> . Notes: <ul style="list-style-type: none"> For multi-word startup options, such as <code>read only</code> and <code>read write</code>, separate the words with a space and enclose in double quotation marks (" "). For example, "read only". When performing a switchover in an Oracle Data Guard configuration, the <code>-startoption</code> for a standby database that becomes a primary database is always set to <code>OPEN</code> after the switchover.
<code>-stopoption</code> <code>stop_options</code>	Specify stop options for the database, such as <code>NORMAL</code> , <code>TRANSACTIONAL</code> , <code>IMMEDIATE</code> , and <code>ABORT</code> .
<code>-dbname</code> <code>db_name</code>	Specify the name of the database, if it is different from the unique name given by the <code>-db</code> parameter.
<code>-acfspath</code> <code>"acfs_path_list"</code>	A single Oracle ACFS path or a comma-delimited list of Oracle ACFS paths enclosed in double quotation marks (" ") where the database's dependency is set. Use this parameter to create dependencies on Oracle ACFS file systems other than <code>ORACLE_HOME</code> , such as for when the database uses <code>ORACLE_BASE</code> on a file system that is different from the <code>ORACLE_HOME</code> file system.
<code>-policy</code> { <code>AUTOMATIC</code> <code>MANUAL</code> <code>NORESTART</code> }	Specify the management policy for the database. <ul style="list-style-type: none"> <code>AUTOMATIC</code> (default): The database is automatically restored to its previous running condition (started or stopped) upon restart of the database host computer. <code>MANUAL</code>: The database is never automatically restarted upon restart of the database host computer. A <code>MANUAL</code> setting does not prevent Oracle Clusterware from monitoring the database while it is running and restarting it if a failure occurs. <code>NORESTART</code>: Similar to the <code>MANUAL</code> setting, the database is never automatically restarted upon restart of the database host computer. A <code>NORESTART</code> setting, however, never restarts the database even if a failure occurs.
<code>-serverpool</code> <code>"server_pool_name" [-</code> <code>ppool "pq_pool_name"]</code>	Specify the name of a server pool used to control database placement. If you do not specify this parameter, then it defaults to the Generic server pool. You can optionally also specify the name of a parallel query server pool to be used by the database. Notes: <ul style="list-style-type: none"> This parameter can only be used with Oracle Clusterware. You can use this parameter with the <code>-node</code> parameter but the server pool must have <code>MAX_SIZE=1</code> and exactly one configured server (the one you specify in <code>-node</code>). After you add server pools, you can assign services to them using the srvctl add service command.
<code>-diskgroup</code> <code>"disk_group_list"</code>	Specify a comma-delimited list of Oracle ASM disk groups if database uses Oracle ASM storage.

Table A-5 (Cont.) srvctl add database Command Parameters

Parameter	Description
<code>-css_critical {YES NO}</code>	You can add weight to a service by setting this parameter to <code>YES</code> . In the event of a node failure within the cluster, Oracle Clusterware will evict the node with the least amount of weight, ensuring that critical services remain available. Note: You can use this parameter only on an administrator-managed node. Should the node become policy managed, at some point, this parameter will no longer apply.
<code>-cpucount <i>cpu_count</i></code>	Specify the number of CPUs. The default value is 0.
<code>-memorytarget <i>memory_target</i></code>	Specify the target memory, in MB, to be allocated for the database. The default is 0.
<code>-maxmemory <i>max_memory</i></code>	Specify the maximum memory, in MB, to be allocated for the resource. If you specify <code>-memorytarget</code> but not <code>-maxmemory</code> , then <code>-maxmemory</code> will be the default value of 0. Both <code>-maxmemory</code> and <code>-memorytarget</code> are validated as long as <code>-memorytarget</code> is less than or equal to <code>-maxmemory</code> .
<code>-cpucap <i>cpu_cap</i></code>	Specify a percentage from 1 to 100 that is the maximum utilization of the workload CPUs the database requires. The default is 0.
<code>-defaultnetnum <i>network_number</i></code>	Specify a network number (an integer) to which services will default in the event you do not specify a network number when you add the service. The number must match the value of the <code>-netnum</code> parameter you specified when you added the network.

Examples

An example of this command to add a policy-managed Oracle RAC database is:

```
$ srvctl add database -db crm -oraclehome /u01/oracle/product/12c/mydb
  -domain example.com -spfile +diskgroup1/crm/spfilecrm.ora
  -role PHYSICAL_STANDBY -startoption MOUNT -dbtype RAC -dbname crm_psd
  -policy MANUAL -serverpool "svrpool1,svrpool2" -diskgroup "dgrp1,dgrp2"
```

An example of this command to add an administrator-managed database is:

```
$ srvctl add database -db crm -oraclehome /u01/oracle/product/12c/mydb
  -domain example.com
```

srvctl config database

Displays the configuration for an Oracle RAC database or lists all configured databases that are registered with Oracle Clusterware.

Syntax

```
srvctl config database [-db db_unique_name] [-all] [-verbose]
```

Parameters

Table A-6 `srvctl config database` Command Parameters

Parameter	Description
<code>-db db_unique_name</code>	Unique name for the database. If you do not specify this parameter, then the utility displays the configuration of all database resources.
<code>-all</code>	Print detailed configuration information.
<code>-verbose</code>	Display verbose output.

Example

This command returns output similar to the following:

```
$ srvctl config database -d main4

Database unique name: main
Database name:
Oracle home: /ade/mjkeenan_main4/oracle
Oracle user: mjkeenan
Spfile:
Password file:
Domain:
Start options: open
Stop options: immediate
Database role: PRIMARY
Management policy: AUTOMATIC
Server pools:
Disk Groups:
Mount point paths:
Services: test
Type: RAC
Start concurrency:
Stop concurrency:
OSDBA group: dba
OSOPER group: oper
Database instances: main41,main42
Configured nodes: mjkeenan_main4_0,mjkeenan_main4_1
CSS critical: no
CPU count: 0
Memory target : 0
Maximum memory: 0
CPU cap: 0
Database is administrator managed
```

srvctl convert database

Converts a database either to or from an Oracle RAC One Node database.

Syntax

Use this command with one of the following syntax models:

```
srvctl convert database -db db_unique_name -dbtype RACONENODE
[-instance instance_name] [-timeout timeout]
```

```
srvctl convert database -db db_unique_name -dbtype RAC [-node node_name]
```

Parameters

Table A-7 srvctl convert database Command Parameters

Parameter	Description
-db <i>db_unique_name</i>	Specify the unique name for the database. Note: If you specify a noncluster database, then command returns an error instructing you to use <code>rconfig</code> to convert the noncluster database to Oracle RAC or Oracle RAC One Node.
-dbtype RACONENODE RAC	Specify the type of database to which you are converting, either Oracle RAC One Node or Oracle RAC. Note: If there is an ongoing or failed online database relocation, then the command returns an error instructing you to first complete or abort the online database relocation and then rerun the command.
-instance <i>instance_name</i>	Optionally, you can specify an instance name prefix for Oracle RAC One Node databases. The default value for this parameter is the first 12 characters of the global unique name of the database. Notes: <ul style="list-style-type: none"> You can use this parameter only when converting from an Oracle RAC database to an Oracle RAC One Node database. In order for the converted instance to come online, you must restart the database using the <code>srvctl stop/start database</code> commands.
-timeout <i>timeout</i>	Optionally, you can specify online database relocation timeout, in minutes, for Oracle RAC One Node databases. The default is 30.
-node <i>node_name</i>	Optionally, you can specify the name of the node for an administrator-managed Oracle RAC database. The default is the first candidate. Note: If you do not specify a node name or you specify a node name where the database is not running, then the command returns an error instructing you specify the correct node.

Example

An example of this command is:

```
$ srvctl convert database -db myDB -dbtype RACONENODE -instance myDB3
```

srvctl disable database

Disables a running database.

If the database is a cluster database, then its instances are also disabled.

Syntax

```
srvctl disable database -db db_unique_name [-node node_name]
```

Parameters

Table A-8 srvctl disable database Command Parameters

Parameter	Description
-db <i>db_unique_name</i>	Specify the name of the database you want to disable.
-node <i>node_name</i>	Optionally, you can specify a node on which you want to disable the database. Note: You can only use this parameter only with Oracle Clusterware.

Example

The following example disables the database `mydb1`:

```
$ srvctl disable database -db mydb1
```

srvctl downgrade database

Downgrades the configuration of a database and its services from its current version to a specific lower version.

Syntax

```
srvctl downgrade database -db db_unique_name -oraclehome Oracle_home  
-targetversion to_version
```

Parameters

Table A-9 srvctl downgrade database Command Parameters

Parameter	Description
-db <i>db_unique_name</i>	Specify the unique name of the database you want to downgrade.
-oraclehome <i>Oracle_home</i>	Specify the path to the Oracle home.
-targetversion <i>to_version</i>	Specify the database version to which you want to downgrade.

srvctl enable database

Enables a cluster database and its instances.

Syntax

```
srvctl enable database -db db_unique_name [-node node_name]
```


Parameters

Table A-10 `srvctl enable database` Command Parameters

Parameter	Description
<code>-db db_unique_name</code>	Specify the unique name of the database you want to enable.
<code>-node node_name</code>	Optionally, you can specify the name of the node on which the database resource resides that you want to enable. Note: You can only use this parameter with Oracle Clusterware.

Example

The following example enables a database named `mydb1`:

```
$ srvctl enable database -db mydb1
```

srvctl getenv database

Displays the values for environment variables associated with a database.

Syntax

```
srvctl getenv database -db db_unique_name [-envs "name_list"]
```

Parameters

Table A-11 `srvctl getenv database` Command Parameters

Parameter	Description
<code>-db db_unique_name</code>	Specify the unique name of the database for which you want to display the environment variable values.
<code>-envs "name_list"</code>	Optionally, you can specify a comma-delimited list of the names of specific environment variables enclosed in double quotation marks (" <code>"</code> ") for which you want to display the values. If you do not use this parameter, then SRVCTL displays the values of all environment variables associated with the database.

Example

The following example displays the environment configuration for a database named `crm`:

```
$ srvctl getenv database -db crm
```

srvctl modify database

Modifies the configuration for a database.

Syntax

```
srvctl modify database -db db_unique_name [-eval] [-dbname db_name]
[-instance instance_name] [-oraclehome oracle_home] [-user user_name]
[-server server_list] [-timeout timeout] [-domain db_domain]
```

```

[-spfile spfile] [-pwfile password_file_path]
[-role {PRIMARY|PHYSICAL_STANDBY|LOGICAL_STANDBY|SNAPSHOT_STANDBY}]
[-startoption start_options] [-stopoption stop_options]
[-policy {AUTOMATIC | MANUAL | NORESTART}]
[-serverpool "server_pool_name" [-node node_name]]
[-pqpools pq_server_pool] [{-diskgroup "diskgroup_list" | -nodiskgroup}]
[-acfspace "acfs_path_list" ] [-css_critical {yes | no}] [-cpucount cpu_count]
[-memorytarget memory_target] [-maxmemory max_memory]
[-cpucap cpu_cap] [-defaultnetnum network_number] [-force]

```

Parameters

Table A-12 srvctl modify database Command Parameters

Parameter	Description
-db <i>db_unique_name</i>	Unique name for the database.
-eval	Use this parameter to hypothetically evaluate the impact of the command on the system. Note: You can only use this parameter with a policy-managed database.
-dbname <i>db_name</i>	The name of the database, if it is different from the unique name given by the -db parameter.
-instance <i>instance_name</i>	Instance name prefix; this parameter is required for administrator-managed Oracle RAC One Node databases.
-oraclehome <i>oracle_home</i>	The path for the Oracle database home directory.
-user <i>user_name</i>	The name of the user that owns the Oracle home directory. Note: If you specify the -user parameter, then you must run this command in privileged mode.
-server <i>server_list</i>	List candidate servers for Oracle RAC One Node databases. Note: You can use this parameter only with administrator-managed Oracle RAC One Node databases. If your Oracle RAC One Node database is policy managed, you <i>cannot</i> use this parameter.
-timeout <i>timeout</i>	Online database relocation timeout, in minutes, for Oracle RAC One Node databases. The default is 30.
-domain <i>db_domain</i>	The domain for the database. Note: You must use this parameter if you set the DB_DOMAIN initialization parameter for the database.
-spfile <i>spfile</i>	The path name of the database server parameter file.
-pwfile <i>password_file_path</i>	Enter the full path to the location of the password file.
-role {PRIMARY PHYSICAL_STANDBY LOGICAL_STANDBY SNAPSHOT_STANDBY}	The role of the database in an Oracle Data Guard configuration. The default is PRIMARY.

Table A-12 (Cont.) srvctl modify database Command Parameters

Parameter	Description
<code>-startoption</code> <code>start_options</code>	<p>Startup options for the database, such as <code>OPEN</code>, <code>MOUNT</code>, and <code>NOMOUNT</code>. The default value is <code>OPEN</code>.</p> <p>Notes:</p> <ul style="list-style-type: none"> For multi-word startup options, such as <code>read only</code> and <code>read write</code>, separate the words with a space and enclose in double quotation marks (" "). For example, "read only". When performing a switch-over in an Oracle Data Guard configuration, the <code>-startoption</code> for a standby database that becomes a primary database is always set to <code>OPEN</code> after the switchover.
<code>-stopoption</code> <code>stop_options</code>	<p>Stop options for the database, such as <code>NORMAL</code>, <code>TRANSACTIONAL</code>, <code>IMMEDIATE</code>, and <code>ABORT</code>.</p>
<code>-policy</code> { <code>AUTOMATIC</code> <code>MANUAL</code> <code>NORESTART</code> }	<p>Management policy for the database.</p> <ul style="list-style-type: none"> <code>AUTOMATIC</code> (default): The database is automatically restored to its previous running condition (started or stopped) upon restart of the database host computer. <code>MANUAL</code>: The database is never automatically restarted upon restart of the database host computer. A <code>MANUAL</code> setting does not prevent Oracle Clusterware from monitoring the database while it is running and restarting it if a failure occurs. <code>NORESTART</code>: Similar to the <code>MANUAL</code> setting, the database is never automatically restarted upon restart of the database host computer. A <code>NORESTART</code> setting, however, never restarts the database even if a failure occurs.
<code>-serverpool</code> <code>"server_pool_name" [-</code> <code>ppool "pq_pool_name"]</code>	<p>Specify the name of a server pool used to control database placement. If you do not specify this parameter, then it defaults to the Generic server pool.</p> <p>You can optionally also specify the name of a parallel query server pool to be used by the database.</p> <p>Notes:</p> <ul style="list-style-type: none"> This parameter can only be used with Oracle Clusterware. You can use this parameter with the <code>-node</code> parameter but the server pool must have <code>MAX_SIZE=1</code> and exactly one configured server (the one you specify in <code>-node</code>). After you add server pools, you can assign services to them using the srvctl add service command.
<code>-node</code> <code>node_name</code>	<p>Node name on which you want to register a noncluster, or single instance, Oracle database.</p> <p>Note: This parameter can be used only with Oracle Clusterware and can be used with the <code>-serverpool</code> parameter to create a policy-managed, noncluster database.</p>
<code>-diskgroup</code> <code>"disk_group_list"</code>	<p>Comma-delimited list of Oracle ASM disk groups if database uses Oracle ASM storage.</p>

Table A-12 (Cont.) srvctl modify database Command Parameters

Parameter	Description
-acfspath "acfs_path_list"	A single Oracle ACFS path or a comma-delimited list of Oracle ACFS paths enclosed in double quotation marks (" ") where the database's dependency is set. Use this parameter to create dependencies on Oracle ACFS file systems other than ORACLE_HOME, such as for when the database uses ORACLE_BASE on a file system that is different from the ORACLE_HOME file system.
-css_critical {YES NO}	You can add weight to a service by setting this parameter to YES. In the event of a node failure within the cluster, Oracle Clusterware will evict the node with the least amount of weight, ensuring that critical services remain available. Note: You can use this parameter only on an administrator-managed node. Should the node become policy managed, at some point, this parameter will no longer apply.
-cpucount <i>cpu_count</i>	Specify the number of CPUs. The default value is 0.
-memorytarget <i>memory_target</i>	Specify the target memory, in MB, to be allocated for the database. The default is 0.
-maxmemory <i>max_memory</i>	Specify the maximum memory, in MB, to be allocated for the resource. If you specify -memorytarget but not -maxmemory, then -maxmemory will be the default value of 0. Both -maxmemory and -memorytarget are validated as long as -memorytarget is less than or equal to -maxmemory.
-cpucap <i>cpu_cap</i>	Specify a percentage from 1 to 100 that is the maximum utilization of the workload CPUs the database requires. The default is 0.
-defaultnetnum <i>network_number</i>	Specify a network number to which services will default in the event you do not specify a network number when you add a service.

 **See Also::**

- *Oracle Data Guard Concepts and Administration* for more information about database roles
- *SQL*Plus User's Guide and Reference* for more information about database startup options
- *SQL*Plus User's Guide and Reference* for more information about database shutdown options

Usage Notes

- The `srvctl modify database` command can convert administrator-managed databases to policy-managed databases. For a running administrator-managed database, if the server list is supplied, then the node where the database is running must be on that list. The instance name prefix cannot be modified after running the `srvctl add database` command.
- You cannot change the management policy from `AUTOMATIC` (using the `-policy` parameter) for Oracle RAC One Node databases. Any attempt to do so results in

an error message. The same is true for the `-node` parameter, which is used to change the node on which a noncluster database runs.

- For policy-managed Oracle RAC One Node databases, you can use the `-serverpool` parameter to move an Oracle RAC One Node database between server pools but you can only specify one server pool. Specifying a list of server pools returns an error.

Examples

The following example changes the role of a database to a logical standby:

```
$ srvctl modify database -db crm -role logical_standby
```

The following example directs the `racTest` database to use the `SYSFILES`, `LOGS`, and `OLTP` Oracle ASM disk groups:

```
$ srvctl modify database -db racTest -diskgroup "SYSFILES,LOGS,OLTP"
```

srvctl predict database

Predicts the consequences of the failure of a specific database.

Syntax

```
srvctl predict database -db db_unique_name [-verbose]
```

Usage Notes

- Specify the unique name of the database you want to check.
- Optionally, you can use the `-verbose` parameter to display detailed output.

srvctl relocate database

Initiates the relocation of an Oracle RAC One Node database from one node to another node.

This command also cleans up after a failed relocation, and you can only use it for relocating Oracle RAC One Node databases.

Syntax

Use this command with one of the following syntax models:

```
srvctl relocate database -db db_unique_name [-node target_node] [-timeout timeout]
  [-stopoption NORMAL] [-verbose]
```

```
srvctl relocate database -db db_unique_name -abort [-revert] [-verbose]
```

Parameters

Table A-13 srvctl relocate database Command Parameters

Parameter	Description
<code>-db <i>db_unique_name</i></code>	Specify the unique name of the database you want to relocate.

Table A-13 (Cont.) srvctl relocate database Command Parameters

Parameter	Description
<code>-node target_node</code>	Optionally, you can specify a target node to which to relocate the Oracle RAC One Node database. Note: You must use this parameter if you are relocating an administrator-managed Oracle RAC One Node database.
<code>-timeout timeout</code>	Optionally, you can specify an online database relocation timeout, in minutes, for Oracle RAC One Node databases. The default is 30.
<code>-stopoption NORMAL</code>	Optionally, you can use this parameter to shut down an existing database instance using <code>SHUTDOWN NORMAL</code> , instead of the default of <code>SHUTDOWN TRANSACTIONAL LOCAL</code> for a primary database or <code>SHUTDOWN IMMEDIATE</code> for a standby database.
<code>-abort</code>	Use this parameter to abort a failed online database relocation.
<code>-revert</code>	Use this parameter to remove the target node of a failed online relocation request from the candidate server list of an administrator-managed Oracle RAC One Node database.
<code>-verbose</code>	Use this parameter to display verbose output.

Usage Notes

- If the Oracle RAC One Node database you want to relocate is not running, then the command returns an error.
- If another online database relocation is active for this Oracle RAC One Node database, then the command returns an error.
- If an online database relocation for this Oracle RAC One Node database has failed and the target nodes are not the same for either relocation, then the command returns an error instructing you to abort the failed online database relocation and then initiate a new one.
- If an online database relocation for this Oracle RAC One Node database has failed and the target nodes are the same (or you do not specify the target), then the command attempts to relocate the database.

Example

The following example relocates an administrator-managed Oracle RAC One Node database named `rac1` to a server called `node7`.

```
$ srvctl relocate database -db rac1 -node node7
```

srvctl remove database

Removes database configurations.

After running this command, ensure that the password file is in the default location if you want to connect to the database as the SYS user with the SYS user's password.

Syntax

```
srvctl remove database -db db_unique_name [-force] [-noprompt] [-verbose]
```

Parameters

Table A-14 `srvctl remove database` Command Parameters

Parameter	Description
<code>-database</code> <i>db_unique_name</i>	Unique name for the database.
<code>-force</code>	Forcibly remove the database and ignore any dependencies.
<code>-noprompt</code>	Suppress prompts.
<code>-verbose</code>	Display verbose output.

Example

To remove a database named `crm`:

```
$ srvctl remove database -db crm
```

srvctl setenv database

Administers cluster database environment configurations.

Syntax

Use this command with one of the following syntax models:

```
srvctl setenv database -db db_unique_name -envs "name=val[,...]"
```

```
srvctl setenv database -db db_unique_name -env "name=val"
```

Parameters

Table A-15 `srvctl setenv database` Command Parameters

Parameter	Description
<code>-db</code> <i>db_unique_name</i>	Specify a unique name for the database for which you want to set environment variables.
<code>-envs</code> "name=val[,...]"	Specify a comma-delimited list of name-value pairs of environment variables enclosed in double quotation marks (") that you want to set.
<code>-env</code> "name=val"	Specify a single environment variable that you want to set to a value that contains commas or other special characters enclosed in double quotation marks (").

Usage Notes

Add additional information about the command here.

Example

The following example sets the language environment variable for a cluster database:

```
$ srvctl setenv database -db crm -env LANG=en
```

srvctl start database

Starts a cluster database and its enabled instances, and all listeners on nodes with database instances.

You can disable listeners that you do not want to start.

Syntax

```
srvctl start database -db db_unique_name [-eval] [-startoption start_options]  
[-stopconcurrency number_of_instances] [-node node_name]
```

Parameters

Table A-16 srvctl start database Command Parameters

Parameter	Description
-db <i>db_unique_name</i>	Specify the unique name of the database you want to start.
-eval	Optionally, use this parameter to hypothetically evaluate the impact of the command on the system.
-startoption <i>start_options</i>	Optionally, you can set options for the startup command (for example: OPEN, MOUNT, or NOMOUNT). Notes: <ul style="list-style-type: none"> This command parameter supports all database startup options. For multi-word startup options, such as <code>read only</code> and <code>read write</code>, separate the words with a space and enclose in double quotation marks (" "). For example, "read only". See Also: <i>SQL*Plus User's Guide and Reference</i> for more information about startup options
-stopconcurrency <i>number_of_instances</i>	Optionally, you can specify a number of database instances to start simultaneously, or specify 0 for an empty start concurrency value.

Table A-16 (Cont.) srvctl start database Command Parameters

Parameter	Description
<code>-node <i>node_name</i></code>	<p>Optionally, you can specify the name of a node on which you want to start the database.</p> <p>Notes:</p> <ul style="list-style-type: none"> This command only applies to Oracle RAC One Node databases. The node you specify must be in the candidate list for an administrator-managed Oracle RAC One Node database or in the server pool for a policy-managed Oracle RAC One Node database. If the database is already running on a node than the one you specify, then the command returns an error. If you do not specify a node, then Oracle Clusterware chooses which node on which to start the Oracle RAC One Node database according to its policies, such as dispersion, number of resources, and order of candidate nodes. If there is an active online database relocation for the Oracle RAC One Node database you are attempting to start, then both instances will already be running and the command returns an error message.. (Only during an online database relocation are two instances of an Oracle RAC One Node database in existence.) <p>If the online database relocation failed for the Oracle RAC One Node database and you do not specify a node, then the command attempts to start both database instances.</p> <p>If the online database relocation failed for the Oracle RAC One Node database and you specify a node, then the command attempts to abort the failed relocation and start the instance on that node.</p>

Examples

The following example starts the `crm` database and sets the startup option to read only:

```
$ srvctl start database -db crm -startoption "read only"
```

srvctl status database

This command displays the current state of the of the database.

Syntax

```
srvctl status database {-db db_unique_name {[-serverpool serverpool_name]  
| [-sid] [-home]} | -serverpool serverpool_name | -thisversion | -thishome}  
[-force] [-detail] [-verbose]
```

Parameters

Table A-17 srvctl status database Parameters

Parameter	Description
<code>-db <i>db_unique_name</i></code>	Specify a unique name for the database.

Table A-17 (Cont.) srvctl status database Parameters

Parameter	Description
-serverpool <i>serverpool_name</i>	Optionally, you can specify a server pool that SRVCTL will display information on nodes contained within.
-sid	Use this parameter to display the SID of the Oracle instance running on this node.
-home	Use this parameter to display the Oracle home of the specified database.
-thisversion	Use this parameter to display the status of databases that are of the same Oracle product version as SRVCTL.
-thishome	Use this parameter to display the status of databases that are configured in this Oracle home.
-force	Include disabled applications
-detail	Use this parameter to display detailed database status information.
-verbose	Displays STATE_DETAILS and INTERNAL_STATE attributes, which include STABLE, STARTING, STOPPING, and CLEANING. If the INTERNAL_STATE is STABLE, then SRVCTL displays no additional information. If the INTERNAL_STATE is STARTING, then SRVCTL displays: Instance {0} is being started If the INTERNAL_STATE is CLEANING, then SRVCTL displays: Instance {0} is being cleaned up

Usage Notes

The output of this command includes information on the Oracle ASM or Oracle ASM IO Server instance for each running instance of the database.

Examples

This command displays output similar to the following:

```
$ srvctl status database -db db00 -detail

Instance {0} is connected to ASM instance {1}
Instance {0} is connected to ASM I/O server instance {1}
```

srvctl stop database

Stops a database, its instances, and its services.

Syntax

```
srvctl stop database -db db_unique_name [-stopoption stop_options]
  [-stopconcurrency number_of_instances] [-drain_timeout timeout] [-eval]
  [-force] [-verbose]
```

Parameters

Table A-18 `srvctl stop database` Command Parameters

Parameter	Description
<code>-db db_unique_name</code>	Specify the unique name for the database that you want to stop.
<code>-stopoption</code> <code>stop_options</code>	Optionally, you can specify options for the shutdown command, such as NORMAL, TRANSACTIONAL LOCAL, IMMEDIATE, OR ABORT.
<code>-stopconcurrency</code> <code>number_of_instances</code>	Optionally, you can specify a number of database instances to stop simultaneously, or specify 0 for an empty stop concurrency value.
<code>-drain_timeout timeout</code>	Optionally, you can specify the time, in seconds, allowed to complete the resource draining action. By default, this parameter is not set. You can specify 0 or any positive integer. An empty string unsets the parameter. If you specify zero, then the agent will perform the actions related to service draining, immediately. Drain timeout is the maximum time the service waits before exiting (in case of <code>srvctl stop service</code> or <code>srvctl stop instance</code>) or proceeding to stop database (<code>srvctl stop database</code>), until the draining of sessions is completed. If session draining completes in 10 seconds and the drain timeout value (on CLI or resource attribute) is 100 seconds, then SRVCTL moves on after 10 seconds. It does not wait for the remaining 90 seconds.
<code>-eval</code>	Optionally, you can use this parameter to hypothetically evaluate the impact of the command on the system.
<code>-force</code>	Optionally, you can use this parameter to stop the database, its instances, its services, and any resources that depend on those services.
<code>-verbose</code>	Optionally, you can use this parameter to display detailed output.

Example

The following command example stops a database and includes detailed output:

```
$ srvctl stop database -db db1 -drain_timeout 50 -verbose
Draining in progress on services svcl,svc2.
Drain complete on services svcl.
Draining in progress on services svc2.
Draining in progress on services svc2.
Drain complete on services svc2.
```

srvctl unsetenv database

Unsets the cluster database environment configurations.

Syntax

```
srvctl unsetenv database -db db_unique_name -envs "name_list"
```

Parameters

Table A-19 `srvctl unsetenv database` Command Parameters

Parameter	Description
<code>-db <i>db_unique_name</i></code>	Specify the unique name of the database for which you want to unset environment variables.
<code>-envs "<i>name_list</i>"</code>	Specify a comma-delimited list of environment variable names enclosed in double quotation marks (" <i>name_list</i> ").

Example

The following example unsets two cluster database environment variables:

```
$ srvctl unsetenv database -db crm -envs "CLASSPATH,LANG"
```

srvctl update database

Updates the specified database to use the new listener endpoints.

Syntax

```
srvctl update database -db db_unique_name
```

Usage Notes

- You can only use this command with Oracle Clusterware.
- Specify the unique name of the database you want to update.

srvctl upgrade database

Upgrades the configuration of a database and all of its services to the version of the database home from where this command is run.

Syntax

```
srvctl upgrade database -db db_unique_name -oraclehome Oracle_home
```

Parameters

Table A-20 `srvctl upgrade database` Command Parameters

Parameter	Description
<code>-db <i>db_unique_name</i></code>	Specify the unique name of the database you want to upgrade.
<code>-oraclehome <i>Oracle_home</i></code>	Specify the path to the upgraded ORACLE_HOME.

srvctl disable diskgroup

Disables a specific disk group on a number of specified nodes.

Syntax

```
srvctl disable diskgroup -diskgroup diskgroup_name [-node "node_list"]
```

Parameters

Table A-21 srvctl disable diskgroup Command Parameters

Parameter	Description
-diskgroup <i>diskgroup_name</i>	Specify the name of the Oracle ASM disk group you want to disable.
-node " <i>node_list</i> "	Optionally, you can specify a comma-delimited list of node names enclosed in double quotation marks (") on which to disable the disk group. Note: You can only use this parameter with Oracle Clusterware.

Example

The following example disables the Oracle ASM disk group, `dgroup1`, on two nodes in a cluster, `mynode1` and `mynode2`:

```
$ srvctl disable diskgroup -diskgroup dgroup1 -node "mynode1,mynode2"
```

srvctl enable diskgroup

Enables a specific disk group on a number of specified nodes.

Syntax

```
srvctl enable diskgroup -diskgroup diskgroup_name [-node "node_list"]
```

Parameters

Table A-22 srvctl enable diskgroup Command Parameters

Parameter	Description
-diskgroup <i>diskgroup_name</i>	Specify the name of the Oracle ASM disk group you want to enable.
-node " <i>node_list</i> "	Optionally, you can specify a comma-delimited list of node names enclosed in double quotation marks (") on which to enable the disk group. Note: You can only use this parameter with Oracle Clusterware.

Example

The following example enables the `diskgroup1` Oracle ASM disk group on nodes `mynode1` and `mynode2`:

```
$ srvctl enable diskgroup -diskgroup diskgroup1 -node "mynode1,mynode2"
```

srvctl predict diskgroup

Predicts the consequences of an Oracle ASM disk group failure.

Syntax

```
srvctl predict diskgroup -diskgroup diskgroup_name [-verbose]
```

Usage Notes

Specify the name of the Oracle ASM disk group for which you want to evaluate a failure. Optionally, you can use the `-verbose` parameter to print detailed output.

srvctl remove diskgroup

Removes a specific Oracle ASM disk group resource from Oracle Clusterware or Oracle Restart.

Syntax

```
srvctl remove diskgroup -diskgroup diskgroup_name [-force]
```

Usage Notes

Specify the name of the Oracle ASM disk group you want to remove. Optionally, you can use the `-force` parameter to ignore any dependencies and forcibly remove the disk group.

Example

The following example forcibly removes the DG1 Oracle ASM disk group:

```
$ srvctl remove diskgroup -diskgroup DG1 -force
```

srvctl start diskgroup

Starts a specific Oracle ASM disk group resource on a number of specified nodes.

Syntax

```
srvctl start diskgroup -diskgroup diskgroup_name [-node "node_list"]
```

Parameters**Table A-23** srvctl start diskgroup Command Parameters

Parameter	Description
<code>-diskgroup <i>diskgroup_name</i></code>	Specify the name of the Oracle ASM disk group you want to start.
<code>-node "<i>node_list</i>"</code>	Optionally, you can specify a comma-delimited list of node names enclosed in double quotation marks (" ") on which to start the disk group resource. Note: You can only use this parameter with Oracle Clusterware.

Example

The following example starts the `diskgroup1` Oracle ASM disk group on the nodes `mynode1` and `mynode2`:

```
$ srvctl start diskgroup -diskgroup diskgroup1 -node "mynode1,mynode2"
```

srvctl status diskgroup

Displays the status of a specific disk group on a number of specified nodes.

Syntax

```
srvctl status diskgroup -diskgroup diskgroup_name [-node "node_list"]
  [-detail] [-verbose]
```

Parameters

Table A-24 srvctl status diskgroup Command Parameters

Parameter	Description
-diskgroup <i>diskgroup_name</i>	Specify the name of the Oracle ASM disk group for which you want to display the status.
-node " <i>node_list</i> "	Optionally, you can specify a comma-delimited list of node names on which to check status of an Oracle ASM disk group. Note: You can only use this parameter with Oracle Clusterware.
-detail	Optionally, you can use this parameter to display detailed status information for the Oracle ASM disk group.
-verbose	Optionally, you can use this parameter to display verbose output.

Examples

The following example displays the status of the `dgrp1` Oracle ASM disk group:

```
$ srvctl status diskgroup -diskgroup dgrp1 -node "mynode1,mynode2" -detail
```

srvctl stop diskgroup

Stops a specific Oracle ASM disk group resource on a number of specified nodes.

Syntax

```
srvctl stop diskgroup -diskgroup diskgroup_name [-node "node_list"] [-force]
```

Parameters

Table A-25 srvctl stop diskgroup Command Parameters

Parameter	Description
-diskgroup <i>diskgroup_name</i>	Specify the name of the Oracle ASM disk group you want to stop.
-node " <i>node_list</i> "	Optionally, you can specify a comma-delimited list of node names enclosed in double quotation marks (" ") on which to stop the Oracle ASM disk group resource. Note: You can only use this parameter with Oracle Clusterware.

Table A-25 (Cont.) srvctl stop diskgroup Command Parameters

Parameter	Description
<code>-force</code>	Optionally, you can use this parameter to perform a forceful dismount. While this parameter does not stop the databases that depend on the disk group you are stopping, it still may cause those databases to fail.

Example

The following command stops the `diskgroup1` Oracle ASM disk group on the two nodes `mynode1` and `mynode2`:

```
$ srvctl stop diskgroup -diskgroup diskgroup1 -node "mynode1,mynode2" -force
```

srvctl start home

Starts all the Oracle Restart-managed or Oracle Clusterware-managed resources on the specified Oracle home.

Syntax

```
srvctl start home -oraclehome Oracle_home -statefile state_file -node node_name
```

Parameters**Table A-26 srvctl start home Command Parameters**

Parameter	Description
<code>-oraclehome</code> <i>Oracle_home</i>	Specify the path to the Oracle home for which you want to start the Oracle Restart or Oracle Clusterware-managed resources.
<code>-statefile</code> <i>state_file</i>	Specify the path to the directory where you want SRVCTL to write the state file.
<code>-node</code> <i>node_name</i>	Specify the name of the node on which the Oracle home resides. Note: You can only use this parameter with Oracle Clusterware.

Example

The following command starts an Oracle home:

```
$ srvctl start home -oraclehome /u01/app/oracle/product/12.2.0/db_1  
-statefile ~/state.txt -node node1
```

srvctl status home

Displays the status of all the Oracle Restart-managed or Oracle Clusterware-managed resources for the specified Oracle home.

Syntax

```
srvctl status home -oraclehome Oracle_home -statefile state_file -node node_name
```


Parameters

Table A-27 `srvctl status home` Command Parameters

Parameter	Description
<code>-oraclehome</code> <i>Oracle_home</i>	Specify the path to the Oracle home for which you want to start the Oracle Restart or Oracle Clusterware-managed resources.
<code>-statefile</code> <i>state_file</i>	Specify the path to the directory that contains the text file that holds the state information generated by this command.
<code>-node</code> <i>node_name</i>	Specify the name of the node on which the Oracle home resides. Note: This parameter is required and you can only use it with Oracle Clusterware.

Example

The following example obtains the status of a particular Oracle home:

```
$ srvctl status home -oraclehome /u01/app/oracle/product/12.1/dbhome_1 -statefile
~/state.txt -node stvm12
```

The preceding command returns output similar to the following:

```
Database cdb1 is running on node stvm12
```

srvctl stop home

Stops all the Oracle Restart-managed or Oracle Clusterware-managed resources that run from the specified Oracle home.

Syntax

```
srvctl stop home -oraclehome Oracle_home -statefile state_file -node node_name
[-stopoption stop_options] [-force]
```

Parameters

Table A-28 `srvctl stop home` Command Parameters

Parameter	Description
<code>-oraclehome</code> <i>Oracle_home</i>	Specify the directory path to the Oracle home for which you want to start the Oracle Restart or Oracle Clusterware-managed resources. Note: The path to the Oracle home you specify must be the same version as the Oracle home from which you invoke SRVCTL.
<code>-statefile</code> <i>state_file</i>	Specify the path to the directory where you want SRVCTL to write the state file.
<code>-node</code> <i>node_name</i>	Specify the name of the node on which the Oracle home resides. Note: You can only use this parameter with Oracle Clusterware.
<code>-stopoption</code> <i>stop_options</i>	Optionally, you can specify shutdown options for the database, such as NORMAL, TRANSACTIONAL, IMMEDIATE, or ABORT See Also: <i>SQL*Plus User's Guide and Reference</i> for more information about shutdown options

Table A-28 (Cont.) srvctl stop home Command Parameters

Parameter	Description
-force	Optionally, you can use this parameter to stop the resources even if errors are reported.

Example

The following example stops the Oracle home:

```
$ srvctl stop home -oraclehome /u01/app/oracle/product/12.1.0/db_1 -statefile
~/state.txt
```

srvctl add instance

Adds a configuration for an instance to your cluster database configuration.

You can only use this command for administrator-managed databases. If you have a policy-managed database, then use the [srvctl modify srvpool](#) command to add an instance to increase either the maximum size, minimum size, or both, of the server pool used by the database.

Syntax

```
srvctl add instance -db db_unique_name -instance instance_name
                    -node node_name [-force]
```

Parameters**Table A-29 srvctl add instance Command Parameters**

Parameter	Description
-db <i>db_unique_name</i>	The unique name of the database you are adding the instance to
-instance <i>instance_name</i>	The name of the instance you are adding
-node <i>node_name</i>	The name of the node on which you are creating the instance
-force	Optionally, you can force the add operation, even though some resources will be stopped.

Usage Notes

- You can only use this command with Oracle Clusterware and Oracle RAC.
- This command increments the `CARDINALITY` resource attribute.
- If you attempt to use this command on an Oracle RAC One Node database, then the command returns an error stating you must convert the database to Oracle RAC.

Examples

Examples of this command are:

```
$ srvctl add instance -db crm -instance crm01 -node gm01
$ srvctl add instance -db crm -instance crm02 -node gm02
$ srvctl add instance -db crm -instance crm03 -node gm03
```

srvctl disable instance

Disables a database instance.

If the database instance that you disable with this command is the last enabled database instance, then this operation also disables the database.

Note:

- This command is only available with Oracle Clusterware and Oracle RAC.
- If you run this command on an Oracle RAC One Node database, then the command returns an error instructing you to use the `database noun`, instead.

Syntax

```
srvctl disable instance -db db_unique_name -instance "instance_name_list"
```

Parameters

Table A-30 srvctl disable instance Command Parameters

Parameter	Description
-db <i>db_unique_name</i>	Specify the unique name for the database for which you want to disable the instance.
-instance " <i>instance_name_list</i> "	Specify an instance name or a comma-delimited list of instance names enclosed in double quotation marks (") you want to disable.

Example

The following example disables two instances of the `crm` database, named `crm1` and `crm2`:

```
$ srvctl disable instance -db crm -instance "crm1,crm3"
```

srvctl enable instance

Enables an instance of an Oracle RAC database.

If you use this command to enable all instances, then the database is also enabled.

 **Note:**

- You can only use this command with Oracle Clusterware and Oracle RAC.
- If you run this command on an Oracle RAC One Node database, then the command returns an error instructing you to use the `database` noun, instead.

Syntax

```
srvctl enable instance -db db_unique_name -instance "instance_name_list"
```

Parameters**Table A-31** `srvctl enable instance` Command Parameters

Parameter	Description
<code>-db <i>db_unique_name</i></code>	Specify the unique name of the database for which you want to enable instances.
<code>-instance "<i>instance_name_list</i>"</code>	Specify a comma-delimited list of instance names enclosed in double quotation marks (" ") that you want to enable.

Example

The following example enables two instances of the `crm` database:

```
$ srvctl enable instance -db crm -instance "crm1,crm2"
```

srvctl modify instance

For an administrator-managed database, this command modifies the configuration for a database instance from its current node to another node. For a policy-managed database, this command defines an instance name to use when the database runs on the specified node.

Syntax

```
srvctl modify instance -db db_unique_name -instance instance_name  
-node node_name
```

Parameters**Table A-32** `srvctl modify instance` Command Parameters

Parameter	Description
<code>-database <i>db_unique_name</i></code>	Specify the unique name for the database.

Table A-32 (Cont.) srvctl modify instance Command Parameters

Parameter	Description
-instance <i>instance_name</i>	Specify the database instance name. Notes: <ul style="list-style-type: none"> If you are modifying a policy-managed database instance, then the instance name must contain an underscore (_), such as <i>pmdb1_1</i>. If you specify an instance name that has never been started before, and is not of the form <i>prefix_number</i>, then you may have to assign an instance number, undo, and redo in the SPFILE.
-node <i>node_name</i>	Name of the node on which to run the instance. You can set the value of this parameter to "" only for a policy-managed database.

Usage Notes

You *cannot* use this command to rename or relocate a running instance.

Examples

The following example changes the configuration of an administrator-managed database, *amdb*, so that the database instance, *amdb1*, runs on the specified node, *mynode*:

```
$ srvctl modify instance -db amdb -instance amdb1 -node mynode
```

The following example causes the policy-managed database *pmdb*, when and if it runs on *mynode*, to use the instance name *pmdb1*:

```
$ srvctl modify instance -db pmdb -instance pmdb1_1 -node mynode
```

The following example removes the directive established by the previous example:

```
$ srvctl modify instance -db pmdb -instance pmdb1_1 -node ""
```

srvctl remove instance

Removes the configurations for an instance of an administrator-managed database. To remove the configurations of a policy-managed database, you must shrink the size of the server pool with the [srvctl modify srvpool](#) command.

Syntax

```
srvctl remove instance -db db_unique_name -instance instance_name  
[-noprompt] [-force]
```

Parameters**Table A-33 srvctl remove instance Command Parameters**

Parameter	Description
-db <i>db_unique_name</i>	Specify the unique name of the administrator-managed database.

Table A-33 (Cont.) srvctl remove instance Command Parameters

Parameter	Description
<code>-instance</code> <i>instance_name</i>	Specify the name of the Instance you want to remove.
<code>-noprompt</code>	Use this parameter to suppress prompts.
<code>-force</code>	Use this parameter to skip checking that the instance is not running, and remove it even though it is running. This parameter also skips checking that the instance has no running services using it, and causes those services to stop before the instance is removed.

Usage Notes

- You can use this command only with Oracle Clusterware and Oracle RAC.
- If you use the `-force` parameter, then any services running on the instance stop. Oracle recommends that you reconfigure services to not use the instance you want to removed as a preferred or available instance before removing the instance.
- If you attempt to use this command on an Oracle RAC One Node database, then the command returns an error stating that cannot remove the instance except by removing the database.

Example

The following example removes the `crm01` database instance from the `crm` database.

```
$ srvctl remove instance -db crm -instance crm01
```

srvctl start instance

Starts instances in the cluster database.

Use the `srvctl start instance` command to start database instances, and all listeners on nodes with database instances

Syntax

Use the `srvctl start instance` command with one of these syntax models:

```
srvctl start instance -db db_unique_name -node node_name
  [-instance "instance_name"] [-startoption start_options]
```

```
srvctl start instance -db db_unique_name -instance "inst_name_list"
  [-startoption start_options]
```

Parameters**Table A-34 srvctl start instance Parameters**

Parameter	Description
<code>-db <i>db_unique_name</i></code>	Unique name for the database.

Table A-34 (Cont.) srvctl start instance Parameters

Parameter	Description
-node <i>node_name</i>	The name of a single node. Note: Use this parameter for policy-managed databases.
-instance { " <i>instance_name</i> " " <i>inst_name_list</i> " }	The name of a single instance or a comma-delimited list of instance names Note: Use this parameter for administrator-managed databases.
-startoption <i>start_options</i>	Options for startup command, such as OPEN, MOUNT, or NOMOUNT) Note: For multi-word startup options, such as read only and read write, separate the words with a space and enclose in double quotation marks (" "). For example, "read only".

Usage Notes

- This command is only available with Oracle Clusterware and Oracle RAC.
- If you run this command on an Oracle RAC One Node database, then the command returns an error instructing you to use the `database` noun, instead.

srvctl status instance

Displays the status of instances.

**Note:**

This command is only available with Oracle Clusterware and Oracle RAC.

srvctl stop instance

The `srvctl stop instance` command stops instances and stops any services running on specified instances.

Syntax

```
srvctl stop instance -db db_unique_name {-node node_name
| -instance "instance_name_list" } [-stopoption stop_options]
[-drain_timeout timeout] [-force] [-failover] [-verbose]
```

Parameters**Table A-35 srvctl stop instance Command Parameters**

Parameter	Description
-db <i>db_unique_name</i>	
-node <i>node_name</i>	
-instance " <i>instance_name_list</i> "	

Table A-35 (Cont.) srvctl stop instance Command Parameters

Parameter	Description
-stopoption <i>stop_options</i>	
-drain_timeout <i>timeout</i>	
-force	
-failover	
-verbose	

Usage Notes

If you run this command on an Oracle RAC One Node database, then the command returns an error instructing you to use the `srvctl stop database` command, instead.

Example

The following command example stops instances in an administrator-managed database and includes verbose output:

```
$ srvctl stop instance -db db1 -instance inst1 -drain_timeout 50 -verbose
Draining in progress on services svcl
Draining in progress on services svcl
Drain complete on services svcl
```

srvctl update instance

The `srvctl update instance` command changes to which Oracle ASM instance or Oracle ASM IO Server the instance connects, or it changes the open mode of the instance.

Syntax

```
srvctl update instance -db db_unique_name [-instance "instance_name_list"
| -node "node_list"] [-startoption start_options] [-targetinstance instance_name]
```

Parameters

Parameter	Description
-db <i>db_unique_name</i>	Specify the unique name of the database.
-instance " <i>instance_name_list</i> " -node " <i>node_list</i> "	Optionally, you can specify a comma-delimited list of instance names or node names that you want to update. If you specify a list of node names, then SRVCTL updates the instances running on those specific nodes.
-startoption <i>start_options</i>	Optionally, you can specify startup options for the database, such as OPEN, MOUNT, or "READ ONLY".
-targetinstance <i>instance_name</i>	Optionally, you can specify a target Oracle ASM or Oracle ASM IO Server instance. Use double quotation marks (") to specify the default target instance)

Examples

An example of this command is:

```
$ srvctl update instance -db qdb -node rws1270051 -targetinstance +IOS2
  relocate qdb instance on rws1270050 to IOServer instance +IOS2
```

srvctl add listener

Adds a listener to every node in a cluster.

Syntax

Use this command with one of the following syntax models.

To create an Oracle Database listener:

```
srvctl add listener [-listener listener_name] [-netnum network_number] [-oraclehome
Oracle_home]
  -user user_name [-endpoints "[TCP:]port_list[:FIREWALL={ON|OFF}][[/IPC:key][/
NMP:pipe_name]
  [/{TCPS|SDP|EXADIRECT}port_list[:FIREWALL={ON|OFF}]]]" [-group group_name]] [-
invitednodes "node_list"]
  [-invitedsubnets "subnet_list"] [-skip]
```

To create an Oracle ASM listener:

```
srvctl add listener [-listener listener_name] -asmlistener [-subnet subnet]
  [-endpoints "[TCP:]port_list[:FIREWALL={ON|OFF}][[/IPC:key][/NMP:pipe_name]
  [/{TCPS|SDP|EXADIRECT}port_list[:FIREWALL={ON|OFF}]]]" [-group group_name]] [-
invitednodes "node_list"]
  [-invitedsubnets "subnet_list"] [-skip]
```

To create a Leaf listener:

```
srvctl add listener [-listener listener_name] -leaflistener [-subnet subnet]
  [-endpoints "[TCP:]port_list[:FIREWALL={ON|OFF}][[/IPC:key][/NMP:pipe_name]
  [/{TCPS|SDP|EXADIRECT}port_list[:FIREWALL={ON|OFF}]]]" [-group group_name]] [-
invitednodes "node_list"]
  [-invitedsubnets "subnet_list"] [-skip]
```

To create a SCAN listener, use the [srvctl add scan_listener](#) command.

Parameters

Table A-36 srvctl add listener Command Parameters

Parameter	Description
-listener <i>listener_name</i>	Specify a listener name. This parameter is optional. If you do not specify this parameter, then the name of the listener defaults to LISTENER for a database listener, LISTENER_ASM for an Oracle ASM listener, and LISTENER_LEAF for a Leaf Node listener.
-netnum <i>network_number</i>	The optional network number from which VIPs are obtained. If not specified, the VIPs are obtained from the same default network from which the nodeapps VIP is obtained. Note: Use this parameter when you add an Oracle Database listener.

Table A-36 (Cont.) `srvctl add listener` Command Parameters

Parameter	Description
<code>-oraclehome</code> <code>oracle_home</code>	Specify an Oracle home for the cluster database. If you do not include this parameter, then SRVCTL uses the Grid home by default. Note: Use this parameter when you add an Oracle Database listener.
<code>-user user_name</code>	Use this parameter to set the user that will run the listener to a less privileged user. Oracle recommends using this parameter to increase security. Notes: <ul style="list-style-type: none"> You must be logged in as <code>root</code> to run this command and specify the <code>-user</code> parameter. Use this parameter when you add an Oracle Database listener. When you use the <code>-user</code> parameter, ensure the following: <ul style="list-style-type: none"> The listener log directory in <code>ORACLE_BASE</code> and the <code>Grid_home/network/admin/user_name</code> directory must both exist on each node before you can use this parameter. Additionally, <code>user_name</code> must have read, write, and execute permission in the directory. The <code>\$ORACLE_HOME/network/admin/endpoints_listener.ora</code> file exists and <code>user_name</code> has read and write permission on it. The <code>\$ORACLE_BASE/diag/tnslsnr/host_name/lower_case_listener_name</code> directory exists and <code>user_name</code> has read, write, and execute permission on it. Before you can use LSNRCTL to manage a listener, you must set <code>TNS_ADMIN</code> to <code>Grid_home/network/admin/user_name</code>.
<code>-endpoints</code> " <code>[TCP:]port_list[:FIREWALL={ON OFF}][/IPC:key] [/NMP:pipe_name][/{TCPS SDP EXADIRECT}port_list[:FIREWALL={ON OFF}]</code>]"	Protocol specifications for the listener. Use <code>port_list</code> to specify a comma-delimited list of TCP ports or listener endpoints. If you do not specify the <code>-endpoints</code> parameter for an Oracle Database listener, then SRVCTL searches for a free port between 1521 and 1540. You can also specify endpoints for TCPS, SDP, and EXADIRECT ports. Note: You can modify this attribute using Online Resource Attribute Modification.
<code>-group group_name</code>	Optionally, you can use the <code>-group</code> parameter with <code>-endpoints</code> to specify a group for the secure endpoint.
<code>-invitednodes</code> " <code>node_list</code> "	Specify a comma-delimited list of node names allowed to register with the listener.
<code>-invitedsubnets</code> " <code>subnet_list</code> "	Specify a comma-delimited list of subnets allowed to register with the listener.
<code>-skip</code>	Indicates you want to skip the checking of ports.
<code>-asmlistener</code>	Specifies the listener type as an Oracle ASM listener. If you do not specify the <code>-listener</code> parameter, then the name of the Oracle ASM listener defaults to <code>LISTENER_ASM</code> . Note: You can only use this parameter with Oracle Clusterware.

Table A-36 (Cont.) srvctl add listener Command Parameters

Parameter	Description
-leaflistener	Specifies the listener type is a Leaf Node listener. If you do not specify the <code>-listener</code> parameter, then the name of the Leaf Node listener defaults to <code>LISTENER_LEAF</code> . Note: You can only use this parameter with Oracle Clusterware.
-subnet <i>subnet</i>	Specifies the subnet to use for an Oracle ASM or Leaf Node listener. Note: You can only use this parameter with Oracle Clusterware.

Usage Notes

You must run this command as `root` user on Linux and UNIX platforms when you specify the `-user` parameter.

Example

The following command adds a listener named `listener112` that is listening on ports 1341, 1342, and 1345 and runs from the Oracle home directory on every node in the cluster:

```
$ srvctl add listener -listener listener112 -endpoints "1341,1342,1345"
-oraclehome /u01/app/oracle/product/12.2.0/db1
```

srvctl config listener

Displays configuration information of a specific listener that is registered with Oracle Clusterware.

Syntax

```
srvctl config listener [-listener listener_name | -asmlistener | -leaflistener]
[-all]
```

Parameters**Table A-37 srvctl config listener Command Parameters**

Parameter	Description
-listener <i>listener_name</i> -asmlistener -leaflistener	The name of a specific listener name or the type of listener (Oracle ASM or Leaf Node). If you do not specify this parameter, then SRVCTL displays the configuration for the default database listener.
-all	Print detailed configuration information.

Example

This command returns output similar to the following:

```
Name: LISTENER_LEAF
Subnet: 10.100.200.195
Type: type
```

```
Owner: scott
Home: Grid_home
End points: TCP:1521
```

srvctl disable listener

Disables a listener resource.

Syntax

```
srvctl disable listener [-listener listener_name] [-node node_name]
```

Parameters

Table A-38 srvctl disable listener Command Parameters

Parameter	Description
-listener <i>listener_name</i>	Optionally, you can specify the name of a particular listener resource. If you do not specify this parameter, then the name of the listener defaults to LISTENER.
-node <i>node_name</i>	Optionally, you can specify the name of a cluster node on which the listener resource you want to disable is running. Note: This parameter is only available with Oracle Clusterware.

Example

The following example disables a listener resource named `listener_crm` on the node `node5`:

```
$ srvctl disable listener -listener listener_crm -node node5
```

srvctl enable listener

Enables a listener resource.

Syntax

```
srvctl enable listener [-listener listener_name] [-node node_name]
```

Parameters

Table A-39 srvctl enable listener Command Parameters

Parameter	Description
-listener <i>listener_name</i>	Optionally, you can specify the name of a listener resource. If you do not use this parameter, then the name of the listener defaults to LISTENER.
-node <i>node_name</i>	Optionally, you can specify the name of a cluster node on which to enable the listener. Note: You can only use this parameter with Oracle Clusterware.

Examples

The following example enables the listener named `listener_crm` on the node named `node5`:

```
$ srvctl enable listener -listener listener_crm -node node5
```

srvctl getenv listener

Displays the environment variables for the specified listener.

Syntax

```
srvctl getenv listener [-listener listener_name] [-envs "name_list"]
```

Parameters

Table A-40 `srvctl getenv listener` Command Parameters

Parameter	Description
<code>-listener listener_name</code>	Optionally, you can specify a listener name for which you want to obtain the environment variables. If you do not use this parameter, then the name of the listener defaults to <code>LISTENER</code> .
<code>-envs "name_list"</code>	Optionally, you can specify a comma-delimited list of the names of environment variables enclosed in double quotation marks (" <code>"</code> "). If you do not use this parameter, then SRVCTL displays the values of all environment variables associated with the listener.

Example

The following example lists all environment variables specified for the default listener:

```
$ srvctl getenv listener
```

srvctl modify listener

Changes several aspects of the listener

Changes the Oracle home directory from which the listener runs, the name of the operating system user who owns Oracle home directory from which the listener runs, the listener endpoints, or the public subnet on which the listener listens, either for the default listener, or a specific listener, that is registered with Oracle Restart or with Oracle Clusterware.

If you want to change the name of a listener, then use the [srvctl remove listener](#) and [srvctl add listener](#) commands.

Syntax

```
srvctl modify listener [-listener listener_name] [-oraclehome oracle_home]
[-user user_name] [-netnum network_number]
[-endpoints "[TCP:]port_list[/IPC:key][NMP:pipe_name][TCPS:s_port][SDP:port]"]
```

Parameters

Table A-41 `srvctl modify listener` Command Parameters

Parameter	Description
<code>-listener</code> <i>listener_name</i>	Optionally, you can enter the name of the listener you want to modify. If you do not use this parameter, then SRVCTL uses the default name, LISTENER.
<code>-oraclehome</code> <i>oracle_home</i>	If you choose to use this parameter, then SRVCTL moves the listener to run from the Oracle home you specify. Note: When you use this parameter, run the command as a privileged user to enable SRVCTL to update resource ownership corresponding to the new ORACLE_HOME owner.
<code>-user</code> <i>user_name</i>	Optionally, you can specify the name of the operating system user who will own the specified Oracle listener Note: You can only use this parameter with Oracle Clusterware.
<code>-netnum</code> <i>network_number</i>	Optionally, you can use this parameter to change the public subnet on which the listener listens. Note: Oracle recommends that you always have at least one listener on the default network. Do not use this parameter to change the network of the only listener that listens on the default network.
<code>-endpoints</code> "[TCP:] <i>port_list</i> [/ IPC: <i>key</i> [/ NMP: <i>pipe_name</i> [/ TCPS: <i>s_port</i> [/ SDP: <i>port</i>]"	Optionally, you can use this parameter to modify protocol specifications for the listener. You must enclose the string of protocols in double quotation marks (""). <i>port_list</i> is comma-delimited list of port numbers. Note: You can modify this attribute using Online Resource Attribute Modification.

Example

The following example changes the TCP ports for the default listener:

```
$ srvctl modify listener -endpoints "TCP:1521,1522"
```

srvctl predict listener

Predicts the consequences of a listener failure.

Syntax

```
srvctl predict listener listener_name [-verbose]
```

Usage Notes

Specify the name of the listener for which you want to predict the consequences of a failure. Optionally, you can use the `-verbose` parameter for detailed output.

srvctl remove listener

Removes the configuration of a specific listener, or all listeners, from Oracle Clusterware or Oracle Restart.

Syntax

```
srvctl remove listener [-listener listener_name | -all] [-force]
```

Usage Notes

- Optionally, you can specify the name of a listener that you want to remove or use the `-all` parameter to remove all listeners. If you do not specify a listener name, then the listener name defaults to `LISTENER` for a database listener; `LISTENER_ASM` for an Oracle ASM listener; or `LISTENER_LEAF` for a Leaf Node listener.
- Optionally, you can use the `-force` parameter to skip checking whether there are other resources that depend on this listener, such as databases, and remove the listener anyway.

Example

The following example removes the configuration for the listener named `lsnr01`:

```
$ srvctl remove listener -listener lsnr01
```

srvctl setenv listener

Administers listener environment configurations.

Syntax

Use this command with one of the following syntax models:

```
srvctl setenv listener [-listener listener_name] -envs "name=val[,...]"
```

```
srvctl setenv listener [-listener listener_name] -env "name=val"
```

Parameters

Table A-42 srvctl setenv listener Command Parameters

Parameter	Description
<code>-listener</code> <i>listener_name</i>	Optionally, you can specify the name of a listener. If you do not use this parameter, then the listener name defaults to <code>LISTENER</code> .
<code>-envs "name=val[,...]"</code>	Specify a comma-delimited list of name-value pairs of environment variables enclosed in double quotation marks (" ").
<code>-env "name=val"</code>	Use this parameter to enable single environment variable to be set to a value that contains commas or other special characters enclosed in double quotation marks (" ").

Examples

The following example sets the language environment configuration for the default listener:

```
$ srvctl setenv listener -env "LANG=en"
```

srvctl start listener

Starts the default listener on specific node, or starts the specified listener on all nodes that are registered with Oracle Clusterware or on the given node.

Syntax

```
srvctl start listener [-node node_name] [-listener listener_name]
```

Parameters

Table A-43 srvctl start listener Command Parameters

Parameter	Description
-node <i>node_name</i>	Specify a particular node name to start the listener on that node. Note: You can only use this parameter with Oracle Clusterware.
-listener <i>listener_name</i>	Specify a particular listener name. Use the srvctl config listener command to obtain the name of a listener. If you do not assign a value to this parameter, then SRVCTL starts all known listeners in the cluster.

Examples

An example of this command is:

```
$ srvctl start listener -listener LISTENER_LEAF
```

srvctl status listener

Displays the status of listener resources.

Syntax

```
srvctl status listener [-listener listener_name] [-node node_name] [-verbose]
```

Parameters

Table A-44 srvctl status listener Command Parameters

Parameter	Description
-listener <i>listener_name</i>	Optionally, you can specify the name of a listener. If you do not use this parameter, then the listener name defaults to LISTENER.
-node <i>node_name</i>	Optionally, you can specify the name of a cluster node. Note: You can only use this parameter with Oracle Clusterware.

Table A-44 (Cont.) srvctl status listener Command Parameters

Parameter	Description
-verbose	Optionally, you can use this parameter to display verbose output.

Examples

The following example displays the status of the default listener on the node `node2`:

```
$ srvctl status listener -node node2
```

srvctl stop listener

Stops the default listener or a specific listener on all nodes or the specified node. You can also use this command to stop a listener on a non-cluster database from the non-cluster database home. However, SRVCTL does not accept the `-node` parameter when run from a non-cluster database home.

Syntax

```
srvctl stop listener [-listener listener_name] [-node node_name] [-force]
```

Parameters**Table A-45 srvctl stop listener Command Parameters**

Parameter	Description
-listener <i>listener_name</i>	Specify the name of the listener you want to stop. If you do not assign a value to this parameter, then SRVCTL stops all known listeners in the cluster.
-node <i>node_name</i>	Optionally, you can specify the name of a single node on which a particular listener runs. Note: You can only use this parameter with Oracle Clusterware.
-force	Forcibly stop the listener.

Examples

The following command stops all listeners on the node `mynode1`:

```
$ srvctl stop listener -node mynode1
```

srvctl unsetenv listener

Unsets the environment configuration for a listener.

Syntax

```
srvctl unsetenv listener [-listener listener_name] -envs "name_list"
```

Parameters

Table A-46 `srvctl unsetenv listener` Command Parameters

Parameter	Description
<code>-listener</code> <code>listener_name</code>	Optionally, you can specify the name of a listener for which you want to unset the environment configuration. If you do not use this parameter, then the name of the listener defaults to <code>LISTENER</code> .
<code>-envs "name_list"</code>	Specify a comma-delimited list of environment variable names enclosed in double quotation marks (") that you want to unset.

Examples

The following example unsets the environment variable `TNS_ADMIN` for the default listener:

```
$ srvctl unsetenv listener -envs "TNS_ADMIN"
```

srvctl update listener

Updates the listener to listen on the new endpoints.

Syntax

```
srvctl update listener
```

Usage Notes

- This command does not accept any additional parameters, except for `-help`.
- You can only use this command with Oracle Clusterware.

srvctl add network

Adds a static or dynamic network.

If your server connects to more than one network, then you can use this command to configure an additional network interface for Oracle RAC, allowing you to create VIPs on multiple public networks.

Syntax

```
srvctl add network [-netnum net_number] -subnet subnet/netmask[/if1[/if2...]]
  [-nettype {static | dhcp | autoconfig | mixed}]
  [-leaf] [-pingtarget "ping_target_list"] [-verbose]
```

Parameters

Table A-47 `srvctl add network` Command Parameters

Parameter	Description
<code>-netnum</code> <i>net_number</i>	Specify a network number. The default is 1.

Table A-47 (Cont.) `srvctl add network` Command Parameters

Parameter	Description
<code>-subnet subnet/netmask [/if1[if2 ...]]</code>	Defines a subnet. If you do not specify any interface names, then the network uses any interface on the given subnet. For IPv6, <i>netmask</i> is a prefix length, such as 64.
<code>-nettype {static dhcp autoconfig mixed}</code>	Specify the network type: <code>static</code> , <code>dhcp</code> , <code>autoconfig</code> , or <code>mixed</code> . If you specify <code>static</code> for the network type, then you must provide the virtual IP address using the <code>srvctl add vip</code> command. If you specify <code>dhcp</code> for the network type, then the VIP agent obtains the IP address from a DHCP server. If you specify <code>autoconfig</code> for the network type, then the VIP agent generates a stateless IPv6 address for the network. You can only use <code>autoconfig</code> for IPv6 networks. If the <code>subnet/netmask</code> specification is not for an IPv6 address, then SRVCTL returns an error. If you specify <code>mixed</code> for the network type, then the VIP resource uses both a static IP address and an IP address obtained dynamically, either from a DHCP server for IPv4 or using stateless autoconfiguration for IPv6.
<code>-leaf</code>	Specify the <code>-leaf</code> parameter when you want to create an application VIP network resource that runs only on Leaf Nodes and is used by application VIPs.
<code>[-pingtarget "ping_target_list"]</code>	Specify a comma-delimited list of IP addresses or host names to ping.
<code>-verbose</code>	Verbose output.

Usage Notes

- 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.
- This command is only available with Oracle Clusterware.
- Oracle only supports DHCP-assigned networks for the default network, not for subsequent networks.
- You can also use the `LISTENER_NETWORKS` database initialization parameter to control client redirects to the appropriate network.

Example

An example of this command is:

```
# srvctl add network -netnum 3 -subnet 192.168.3.0/255.255.255.0
```

srvctl config network

Displays the network configuration for the cluster.

Syntax

```
srvctl config network [-netnum network_number]
```

Usage Notes

- Specify the network for which you want to display configuration information.

- This command is only available with Oracle Clusterware.

Example

An example of this command is:

```
$ srvctl config network -netnum 2
```

srvctl modify network

Modifies the subnet, network type, or IP address type for a network.

Syntax

```
srvctl modify network [-netnum network_number] [-subnet subnet/netmask
  [/if1[/if2...]]] [-nettype network_type | -iptype {ipv4 | ipv6 | both}]
  [-pingtarget "ping_target_list"] [-verbose]
```

Parameters

Table A-48 srvctl modify network Command Parameters

Parameter	Description
<code>-netnum <i>network_number</i></code>	Optionally, you can specify a network number that you want to modify. The default is 1.
<code>-subnet <i>subnet/netmask</i> [/<i>if1</i>[/<i>if2</i>...]]</code>	Optionally, you can specify a subnet number for the public network. The netmask and interfaces you specify, if any, change those of the network you are modifying. If you specify an IPv6 subnet, then enter a prefix length, such as 64, in place of <i>netmask</i> . If you do not specify any interface names, then the VIPs use any interface on the given subnet. If you are changing the network type using the <code>-nettype</code> parameter, then you must specify either an existing IPv4 or IPv6 network using the <code>-subnet</code> parameter. Additionally, the subnet and netmask you specify in the <code>-subnet</code> parameter do not change those of the network you are modifying.
<code>-nettype <i>network_type</i></code>	Optionally, you can modify the network type using this parameter, to <i>static</i> , <i>dhcp</i> , <i>autoconfig</i> , or <i>mixed</i> .
<code>-iptype {<i>ipv4</i> <i>ipv6</i> <i>both</i>}</code>	Alternative to modifying the network type, you can modify the type of IP address to <i>ipv4</i> , <i>ipv6</i> , or <i>both</i> .
<code>-pingtarget "<i>ping_target_list</i>"</code>	Optionally, you can specify a comma-delimited list of IP addresses or host names to ping enclosed in double quotation marks ("").
<code>-verbose</code>	Optionally, you can use this parameter to display detailed output.

Usage Notes

- You can only use this command with Oracle Clusterware.
- 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.
- You can modify the IP address type for a network from IPv4 to IPv6, or from IPv6 to IPv4.
- If you specify `static` for the network type, then you must provide the virtual IP address using the `srvctl add vip` command.

- If you specify `dhcp` for the network type, then the VIP agent obtains the IP address from a DHCP server.
- If you specify `autoconfig` for the network type, then the VIP agent generates a stateless IPv6 address for the network. You can only use this parameter for IPv6 networks. If the subnet/netmask specification is not for an IPv6 address, then SRVCTL returns an error.
- If you change a network from `static` to `mixed`, then you must first configure GNS, so that the dynamic addresses obtained can have names registered for them.
- If you specify `mixed` for the network type, then the VIP resource uses both a static IP address and an IP address obtained dynamically, either DHCP or autoconfig.
- If you specify `mixed_autoconfig` for the network type, then the VIP resource retains the static IP configuration and either obtains an IP address from a DHCP server for an IPv4 network specification or generates a stateless auto-configured IP address for an IPv6 network specification.

Examples

The following example changes the subnet number, netmask, and interface list:

```
# srvctl modify network -subnet 192.168.2.0/255.255.255.0/eth0
```

The following example changes the second network to DHCP:

```
# srvctl modify network -netnum 2 -nettype dhcp
```

The following example adds an IPv6 subnet and netmask to the default network:

```
# srvctl modify network -subnet 2606:b400:400:18c0::/64
```

The following example removes the IPv4 configuration from a network:

```
# srvctl modify network -iptype ipv6
```

srvctl predict network

Predicts the consequences of network failure.

Syntax

```
srvctl predict network [-netnum network_number] [-verbose]
```

Usage Notes

Optionally, you can specify a network for which you want to evaluate a failure. The default value is 1. You can also use the `-verbose` parameter to print detailed output.

Example

The following example predicts the consequences of a failure on network number 2:

```
$ srvctl predict network -netnum 2
```

srvctl remove network

Removes the network configuration.

Syntax

```
srvctl remove network {-netnum network_number | -all} [-force] [-verbose]
```

Parameters

Table A-49 srvctl remove network Command Parameters

Parameter	Description
-netnum <i>network_number</i> -all	Specify which network number you want to remove. Alternatively, you can use the -all parameter to indicate that you want to remove all networks.
-force	Optionally, you can use this parameter to remove the specified network regardless of any dependencies.
-verbose	Optionally, you can use this parameter to display detailed output.

Usage Notes

- You can only use the command with Oracle Clusterware.
- 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.

Example

The following example removes a network:

```
# srvctl remove network -netnum 3
```

srvctl add nodeapps

Adds a node application configuration to the specified node.

Syntax

Use this command with one the following syntax models, specifying either a specific node and VIP or a specific subnet and netmask:

```
srvctl add nodeapps
  {-node node_name -address {vip_name | ip_address}/netmask[/if1[/if2[...]]} [-skip]}
  [-emport em_port] [-onslocalport ons_local_port]
  [-onsremoteport ons_remote_port] [-onshostport hostname_port_list]
  [-remoteservers hostname_port_list] [-verbose]
```

```
srvctl add nodeapps -subnet subnet/netmask[/if1[/if2[...]]] [-emport em_port]
  [-onslocalport ons_local_port] [-onsremoteport ons_remote_port]
  [-onshostport hostname_port_list] [-remoteservers hostname_port_list]
  [-verbose]
```

Parameters

Table A-50 `srvctl add nodeapps` Command Parameters

Parameter	Description
<code>-node node_name</code>	The name of the node on which you want to create the node application. Node name is optional and unnecessary if you run the command on the local node.
<code>-address {vip_name ip_address}/netmask[/if1[if2 ...]]</code>	This specification creates a traditional VIP node application on the specified node. Note: You must use this parameter for upgrade configurations and new, non-DHCP configurations.
<code>-skip</code>	Specify this parameter to skip checking the reachability of the VIP address.
<code>-subnet subnet/netmask [/if1[if2 ...]]</code>	Creates a DHCP subnet. If you do not specify any interface names, then the VIPs use any interface on the given subnet.
<code>-emport em_port</code>	Local port on which Oracle Enterprise Manager listens. The default port is 2016.
<code>-onslocalport ons_local_port</code>	The Oracle Notification Service daemon listener port on its node. If you do not specify this value, the Oracle Notification Service daemon listener port defaults to 6100. Note: The local port and remote port must each be unique.
<code>-onsremoteport ons_remote_port</code>	The port number for remote Oracle Notification Service daemon connections. If you do not specify a port number, the default value of 6200 is used for the Oracle Notification Service remote port. Note: The local port and remote port must each be unique.
<code>-onshostport host_port_list</code>	A list of <code>host[:port]</code> pairs of remote hosts that are part of the Oracle Notification Service network but are not part of the Oracle Clusterware cluster Note: If <code>port</code> is not specified for a remote host, then <code>ons_remote_port</code> is used.
<code>-remoteservers host_port_list</code>	A list of <code>host[:port]</code> pairs for Oracle Notification Service daemons on servers that are not in the cluster.
<code>-verbose</code>	Verbose output

Usage Notes

- 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.
- This command is only available with Oracle Clusterware.

Example

An example of this command is:

```
# srvctl add nodeapps -node crmnode1 -address 1.2.3.4/255.255.255.0
```

`srvctl config nodeapps`

Displays the VIP configuration for each node in the cluster.

 **Note:**

This command is only available with Oracle Clusterware.

Syntax

```
srvctl config nodeapps [-viponly] [-onsonly]
```

Usage Notes

Use `-viponly` to display the VIP address configuration. Use `-onsonly` to display the Oracle Notification Service configuration.

Example

An example of this command is:

```
$ srvctl config nodeapps -viponly -onsonly
```

srvctl disable nodeapps

Disables node applications on all nodes in the cluster.

Syntax

```
srvctl disable nodeapps [-gsdonly] [-adminhelper] [-verbose]
```

Parameters**Table A-51** `srvctl disable nodeapps` Command Parameters

Parameter	Description
<code>-gsdonly</code>	Optionally, you can use this parameter to disable only the global services daemon (GSD).
<code>-adminhelper</code>	Optionally, you can use this parameter to disable the Administrator helper only.
<code>-verbose</code>	Optionally, you can use this parameter to display detailed output.

Usage Notes

You can only use this parameter with Oracle Clusterware.

Example

The following example disables GSD:

```
$ srvctl disable nodeapps -gsdonly -verbose
```

srvctl enable nodeapps

Enables the node applications on all nodes in the cluster.

Syntax

```
srvctl enable nodeapps [-gsdonly] [-adminhelper] [-verbose]
```

Parameters

Table A-52 srvctl enable nodeapps Command Parameters

Parameter	Description
-gsdonly	Optionally, you can use this parameter to enable only the global services daemon (GSD).
-adminhelper	Optionally, you can use this parameter to enable the Administrator helper only.
-verbose	Optionally, you can use this parameter to display detailed output.

Usage Notes

You can only use this command with Oracle Clusterware.

Example

The following example enables GSD:

```
$ srvctl enable nodeapps -gsdonly -verbose
```

srvctl getenv nodeapps

Displays the environment variables for the node application configurations.

Syntax

```
srvctl getenv nodeapps [-viponly] [-ononly] [-envs "name_list"]
```

Parameters

Table A-53 srvctl getenv nodeapps Command Parameters

Parameter	Description
-viponly	Optionally, you can use this parameter to display the VIP address configuration.
-ononly	Optionally, you can use this parameter to display the Oracle Notification Service configuration.
-envs "name_list"	Optionally, you can specify a comma-delimited list of the names of environment variables enclosed in double quotation marks (" "). If you do not use this parameter, then SRVCTL displays the values of all environment variables associated with the node applications.

Usage Notes

You can only use this command with Oracle Clusterware.

Example

The following example lists all environment variables for the node applications:

```
$ srvctl getenv nodeapps -viponly
```

srvctl modify nodeapps

Modifies the configuration for a node application.

Syntax

Use this command with one of the following syntax models, specifying either a specific node and VIP or a specific subnet and netmask:

```
srvctl modify nodeapps {[-node node_name -address {vip_name|vip_address}/  
  netmask[/if1[/if2|...]] [-skip]} [-nettype network_type] [-emport em_port]  
  [-onslocalport ons_local_port] [-onsremoteport ons_remote_port]  
  [-remoteservers host:port][,host:port,...] [-verbose]  
  [-clientdata file] [-pingtarget "ping_target_list"]
```

```
srvctl modify nodeapps [-subnet subnet/netmask[/if1[/if2|...]]]  
  [-nettype network_type] [-emport em_port]  
  [-onslocalport ons_local_port] [-onsremoteport ons_remote_port]  
  [-remoteservers host:port][,host:port,...] [-verbose]  
  [-clientdata file] [-pingtarget "ping_target_list"]
```

Parameters

Table A-54 srvctl modify nodeapps Command Parameters

Parameter	Description
-node <i>node_name</i>	Specify the name of the node on which the node application you want to modify resides.
-address { <i>vip_name</i> <i>vip_address</i> }/ <i>netmask</i> [/ <i>if1</i> [/ <i>if2</i> ...]]	Specify a node-level virtual IP name or address. The address specified by name or IP must match the subnet number of the default network. Note: You must use this parameter for upgrade configurations and new non-DHCP configurations
-skip	Optionally, you can use this parameter to skip checking the reachability of the VIP address.
-subnet <i>subnet</i> / <i>netmask</i> [/ <i>if1</i> [/ <i>if2</i> ...]]	Alternative to specifying a node name and address, you can specify a subnet number for the public network. The netmask and interfaces you specify, if any, change those of the default network. Additionally, if you specify a value for the <i>netmask</i> option, then you need only specify it for the first node on each network.
-nettype <i>network_type</i>	Optionally, you can change the network server type to <i>static</i> , <i>dhcp</i> , or <i>mixed</i> .
-emport <i>em_port</i>	Optionally, you can change the local port on which Oracle Enterprise Manager listens. Note: You can also modify this attribute using Online Resource Attribute Modification.

Table A-54 (Cont.) srvctl modify nodeapps Command Parameters

Parameter	Description
<code>-onslocalport</code> <code>ons_local_port</code>	Optionally, you can change the port on which the Oracle Notification Service daemon listens for local client connections. Notes: <ul style="list-style-type: none"> The local port and remote port must each be unique. You can modify the local port while the resource remains online, without restarting the resource.
<code>-onsremoteport</code> <code>ons_remote_port</code>	Optionally, you can change the port on which the Oracle Notification Service daemon listens for connections from remote hosts. Notes: <ul style="list-style-type: none"> The local port and remote port must each be unique. You can modify the remote port while the resource remains online, without restarting the resource.
<code>-remoteservers host:</code> <code>[port][, ...]</code>	Optionally, you can modify the comma-delimited list of <code>host:[port]</code> pairs of remote hosts that are part of the Oracle Notification Service network but are not part of the cluster. If you do not specify a port for a remote host, then the utility uses the value you specified for <code>ons_remote_port</code> .
<code>-clientdata file</code>	Optionally, you can specify the file with a wallet to import, or an empty string to delete a wallet used for SSL to secure Oracle Notification Service communication.
<code>-pingtarget</code> <code>"ping_target_list"</code>	Optionally, you can specify a comma-delimited list of IPs or host names enclosed in double quotation marks (" ") to ping.
<code>-verbose</code>	Optionally, you can use this parameter to display detailed output.

Usage Notes

You can only use this command with Oracle Clusterware.

Example

The following example changes the nodeapps resource on `mynode1` to use the application VIP of `100.200.300.40` with a subnet mask of `255.255.255.0` on the network interface `eth0`:

```
$ srvctl modify nodeapps -node mynode1 -addr 100.200.300.40/255.255.255.0/eth0
```

srvctl remove nodeapps

Removes the node application configuration.

Syntax

```
srvctl remove nodeapps [-force] [-noprompt] [-verbose]
```

Parameters

Table A-55 `srvctl remove nodeapps` Command Parameters

Parameter	Description
<code>-force</code>	Optionally, you can use this parameter to forcibly remove node application configurations, regardless of any dependencies.
<code>-noprompt</code>	Optionally, you can use this parameter to suppress prompts.
<code>-verbose</code>	Optionally, you can use this parameter to display detailed output.

Usage Notes

- You can only use this command with Oracle Clusterware.
- 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.

`srvctl setenv nodeapps`

Sets the environment variables for the node application configurations.

Syntax

```
srvctl setenv nodeapps {-envs "name=val[,...]" | -env "name=val"}
    [-viponly] [-gsdonly] [-ononly] [-verbose]
```

Parameters

Table A-56 `srvctl setenv nodeapps` Command Parameters

Parameter	Description
<code>-envs "name=val[,...]"</code>	Use this parameter to specify a comma-delimited list of name-value pairs of environment variables enclosed in double quotation marks ("").
<code>-env "name=val"</code>	Alternatively, you can use this parameter to enable a single environment variable that is set to a value which contains commas or other special characters, enclosed in double quotation marks ("").
<code>-viponly</code>	Optionally, you can use this parameter to modify only the VIP configuration.
<code>-gsdonly</code>	Optionally, you can use this parameter to modify only the GSD configuration.
<code>-ononly</code>	Optionally, you can use this parameter to modify only the ONS daemon configuration.
<code>-verbose</code>	Optionally, you can use this parameter to display detailed output.

Usage Notes

You can only use this command with Oracle Clusterware.

Example

The following example sets the `CLASSPATH` environment variable for all node applications:

```
$ srvctl setenv nodeapps -env "CLASSPATH=/usr/local/jdk/jre/rt.jar" -verbose
```

srvctl start nodeapps

Starts node-level applications on a node or all nodes in the cluster.

Syntax

```
srvctl start nodeapps [-node node_name] [-gsdonly] [-adminhelper] [-verbose]
```

Parameters**Table A-57 srvctl start nodeapps Command Parameters**

Parameter	Description
-node <i>node_name</i>	Optionally, you can specify a node on which to start node-level applications. If you do not use this parameter, then SRVCTL starts the node applications on all active nodes in the cluster.
-gsdonly	Optionally, you can use this parameter to start only GSD, instead of all node applications.
-adminhelper	Optionally, you can use this parameter to start only an Administrator helper instead of all node applications.
-verbose	Optionally, you can use this parameter to display detailed output.

Usage Notes

You can only use this command with Oracle Clusterware.

srvctl status nodeapps

Displays the status of node applications.

Syntax

```
srvctl status nodeapps [-node node_name]
```

Usage Notes

- You can only use this command with Oracle Clusterware.
- Optionally, you can specify a node for which to display the status of the node applications.

srvctl stop nodeapps

Stops node-level applications on a node in the cluster.

Syntax

```
srvctl stop nodeapps [-node node_name] [-gsdonly] [-adminhelper] [-force]
  [-relocate] [-verbose]
```

Parameters

Table A-58 srvctl stop nodeapps Command Parameters

Parameter	Description
-node <i>node_name</i>	Optionally, you can use this parameter to specify a node on which you want to stop node applications. If you do not use this parameter, then the utility stops the node applications on all active nodes in the cluster.
-gsdonly	Optionally, you can use this parameter to stop only the GSD, instead of all node applications.
-adminhelper	Optionally, you can use this parameter to stop only the Administrator helper instead of all node applications.
-force	Optionally, you can use this parameter to stop node applications regardless of any dependencies.
-relocate	Optionally, you can use this parameter to relocate the VIP and possibly-dependent services. Note: If you use this parameter, then you must also specify the -node <i>node_name</i> parameter.
-verbose	Optionally, you can use this parameter to display detailed output.

Usage Notes

You can only use this command with Oracle Clusterware.

srvctl unsetenv nodeapps

Unsets the environment configuration for the node applications.

Syntax

```
srvctl unsetenv nodeapps -envs "name_list" [-viponly] [-gsdonly] [-ononly]
  [-verbose]
```

Parameters

Table A-59 srvctl unsetenv nodeapps Command Parameters

Parameter	Description
-envs " <i>name_list</i> "	Specify a comma-delimited list of the names of environment variables enclosed in double quotation marks (") that you want to unset.
-viponly	Optionally, you can use this parameter to unset only the VIP configuration.
-gsdonly	Optionally, you can use this parameter to unset only the GSD configuration.

Table A-59 (Cont.) srvctl unsetenv nodeapps Command Parameters

Parameter	Description
-onsonly	Optionally, you can use this parameter to unset only the ONS daemon configuration.
-verbose	Optionally, you can use this parameter to display detailed output.

Example

The following example unsets the environment configuration for the specified node applications:

```
$ srvctl unsetenv nodeapps -envs "test_var1,test_var2"
```

srvctl add ons

Adds an Oracle Notification Service daemon to an Oracle Restart configuration.

Syntax

```
srvctl add ons [-onslocalprt ons_local_port] [-onsremoteport ons_remote_port]  
[-remoteservers host[:port][,...]] [-verbose]
```

Parameters**Table A-60 srvctl add ons Command Parameters**

Parameter	Description
-onslocalprt <i>ons_local_port</i>	Optionally, you can specify the Oracle Notification Service daemon listening port for local client connections. Note: The local port and remote port must each be unique.
-onsremoteport <i>ons_remote_port</i>	Optionally, you can specify the Oracle Notification Service daemon listening port for connections from remote hosts. Note: The local port and remote port must each be unique.
-remoteservers <i>host[:port]</i> [,...]	Optionally, you can specify a comma-delimited list of <i>host:port</i> pairs of remote hosts that are part of the Oracle Notification Service network but are not part of the Oracle Clusterware cluster. Note: If <i>port</i> is not specified for a remote host, then <i>ons_remote_port</i> is used.
-verbose	Optionally, you can use this parameter to display detailed output.

Usage Notes

You can only use this command with Oracle Restart.

Example

An example of this command is:

```
$ srvctl add ons -onslocalprt 6200
```

srvctl config ons

Displays configuration information for the Oracle Notification Service daemon.

Syntax

```
srvctl config ons
```

Usage Notes

- You can only use this command with Oracle Restart.
- This command has no parameters.

srvctl disable ons

Disables the Oracle Notification Service daemon for Oracle Restart installations.

Syntax

```
srvctl disable ons [-verbose]
```

Usage Notes

Optionally, you can use the `-verbose` parameter to display detailed output.

srvctl enable ons

Enables the Oracle Notification Service daemon.

Syntax

```
srvctl enable ons [-verbose]
```

Usage Notes

- You can only use this command with Oracle Restart.
- Optionally, you can use the `-verbose` parameter to display detailed output.

srvctl modify ons

Modifies the ports used by the Oracle Notification Service daemon that is registered with Oracle Restart.

Syntax

```
srvctl modify ons [-onslocalprt ons_local_port] [-onsremoteport ons_remote_port]  
[-emport em_port] [-remoteservers host[:port][,...]] [-verbose]
```


Parameters

Table A-61 `srvctl modify ons` Command Parameters

Parameter	Description
<code>-onslocalprt</code> <code>ons_local_port</code>	Optionally, you can modify the Oracle Notification Service daemon listening port for local client connections. Note: The local port and remote port must each be unique.
<code>-onsremoteport</code> <code>ons_remote_port</code>	Optionally, you can modify the Oracle Notification Service daemon listening port for connections from remote hosts. Note: The local port and remote port must each be unique.
<code>-empport</code> <code>em_port</code>	Optionally, you can specify the local port on which Oracle Enterprise Manager listens. The default port is 2016.
<code>-remoteservers</code> <code>host[:port][,...]</code>	Optionally, you can specify a list of <code>host:port</code> pairs of remote hosts that are part of the Oracle Notification Service network but are not part of the Oracle Clusterware cluster. Note: If you do not specify <code>port</code> for a remote host, then SRVCTL uses the value for <code>ons_remote_port</code> .
<code>-verbose</code>	Optionally, you can use this parameter to display detailed output.

Usage Notes

You can only use this command with Oracle Restart.

`srvctl remove ons`

Removes Oracle Notification Service from the Oracle Grid Infrastructure home.

Syntax

```
srvctl remove ons [-force] [-verbose]
```

Usage Notes

- You can only use this command with Oracle Restart.
- Optionally, you can use the `-force` parameter to remove Oracle Notification Service regardless of dependencies.
- Optionally, you can use the `-verbose` parameter to display detailed output.

`srvctl start ons`

Starts the Oracle Notification Service daemon.

Syntax

```
srvctl start ons [-verbose]
```

Usage Notes

- You can only use this command with Oracle Restart.
- Optionally, you can use the `-verbose` parameter to display detailed output.

srvctl status ons

Displays the current state of the Oracle Notification Service daemon.

Syntax

```
srvctl status ons
```

Usage Notes

- You can only use this command with Oracle Restart.
- This command takes no parameters.

srvctl stop ons

Stops the Oracle Notification Service daemon.

Syntax

```
srvctl stop ons [-verbose]
```

Usage Notes

- You can only use this command with Oracle Restart.
- Optionally, you can use the `-verbose` parameter to display detailed output.

srvctl add scan

Adds Oracle Clusterware resources for the given SCAN.

Syntax

```
srvctl add scan -scanname scan_name [-netnum network_number]
```

Parameters

Table A-62 srvctl add scan Command Parameters

Parameter	Description
<code>-scanname <i>scan_name</i></code>	A fully-qualified host name, which includes the domain name. If the network is dynamic, then you do not have to use fully-qualified host name but, if you choose to do so, then the domain must be the GNS subdomain. Note: You can modify this attribute using Online Resource Attribute Modification.
<code>-netnum <i>network_number</i></code>	The optional network number from which SCAN VIPs are obtained. If you do not specify this parameter, then the SCAN VIPs are obtained from the same default network from which the <code>nodeapps</code> VIP is obtained.

Usage Notes

- This command creates the same number of SCAN VIP resources as the number of IP addresses that SCAN resolves to, or 3 when `network_number` identifies a [dynamic network](#) and Oracle GNS configuration.

- For static networks, the addresses to which the SCAN resolves in DNS must match the address type of the subnet.
- For an IPv4 network, the SCAN must resolve to IPv4 addresses.
- This command is only available with Oracle Clusterware.

Example

An example of this command is:

```
# srvctl add scan -scanname scan.mycluster.example.com
```

srvctl config scan

Displays the configuration information for all SCAN VIPs, by default, or a specific SCAN VIP identified by *ordinal_number*.

Syntax

```
srvctl config scan [[-netnum network_number] [-scannumber ordinal_number] | -all]
```

Parameters

Table A-63 srvctl config scan Command Parameters

Parameter	Description
-netnum <i>network_number</i>	Use this parameter to view the configuration of a specific SCAN VIP.
-scannumber <i>ordinal_number</i>	Use this parameter to specify any one of the three SCAN VIPs, using values from 1 to 3, for which you want to view the configuration.
-all	Alternative to specifying network or ordinal numbers, you can use this parameter to view the configuration for all of the SCAN VIPs.

Usage Notes

This command is only available with Oracle Clusterware.

Example

This command returns output similar to the following:

```
$ srvctl config scan -scannumber 1

SCAN name: mjk12700890090-r, Network: 1
Subnet IPv4: 198.51.100.1/203.0.113.46/eth0, static
Subnet IPv6:
SCAN 1 IPv4 VIP: 198.51.100.195
SCAN VIP is enabled.
SCAN VIP is individually enabled on nodes:
SCAN VIP is individually disabled on nodes:
```

srvctl disable scan

Disables all SCAN VIPs, by default, or a specific SCAN VIP identified by *ordinal_number*.

Syntax

```
srvctl disable scan [-scannumber ordinal_number]
```

Usage Notes

- You can only use this command with Oracle Clusterware.
- Optionally, you can use the `-scannumber` parameter to specify any one of the three SCAN VIPs you want to disable. The parameter takes a range of values from 1 to 3.

Example

The following example disables the first SCAN VIP:

```
$ srvctl disable scan -scannumber 1
```

srvctl enable scan

Enables all SCAN VIPs, by default, or a specific SCAN VIP identified by its ordinal number.

Syntax

```
srvctl enable scan [-scannumber ordinal_number]
```

Usage Notes

- You can only use this command with Oracle Clusterware.
- Optionally, you can use the `-scannumber` parameter to specify any one of the three SCAN VIPs you want to enable. The parameter takes a range of values from 1 to 3.

Example

The following example enables the first SCAN VIP:

```
$ srvctl enable scan -scannumber 1
```

srvctl modify scan

Modifies the number of SCAN VIPs to match the number of IP addresses returned by looking up the *scan_name* you specify in DNS.

You use this command when DNS was modified to add, change, or remove IP addresses, and now you must adjust the Oracle Clusterware resource configuration to match.

Syntax

```
srvctl modify scan -scanname scan_name [-netnum network_number]
```

Parameters

Table A-64 `srvctl modify scan` Command Parameters

Parameter	Description
<code>-scanname scan_name</code>	Identifies the SCAN name that resolves to the SCAN VIPs that you want to modify. Note: You can modify this attribute using Online Resource Attribute Modification.
<code>-netnum network_number</code>	The optional network number from which VIPs are obtained. If not specified, the VIPs are obtained from the same default network from which the <code>nodeapps</code> VIP is obtained.

Example

Assume your system currently has a SCAN named `scan_name1`, and it resolves to a single IP address in DNS. If you modify the SCAN `scan_name1` in DNS to resolve to three IP addresses, then use the following command to create the additional SCAN VIP resources:

```
$ srvctl modify scan -scanname scan_name1
```

srvctl predict scan

Predicts the consequences of SCAN failure.

Syntax

```
srvctl predict scan -scannumber ordinal_number [-verbose]
```

Usage Notes

- Specify an ordinal number that identifies the SCAN VIP for which you want to simulate failure. The range of values you can specify for this parameter is 1 to 3.
- Optionally, you can use the `-verbose` parameter to display detailed output.

Add additional information about the command here.

srvctl relocate scan

Relocates a specific SCAN VIP from its current hosting node to another node within the cluster.

Syntax

```
srvctl relocate scan -scannumber ordinal_number [-node node_name]
```

Parameters

Table A-65 `srvctl relocate scan` Command Parameters

Parameter	Description
<code>-scannumber</code> <i>ordinal_number</i>	Specify an ordinal number that identifies which SCAN VIP you want to relocate. The range of values you can specify for this parameter is 1 to 3.
<code>-node</code> <i>node_name</i>	Optionally, you can specify the name of a single node to which SRVCTL relocates the SCAN VIP. If you do not use this parameter, then SRVCTL chooses the node to which the SCAN VIP is relocated.

Usage Notes

You can only use this command with Oracle Clusterware.

Example

The following example relocates the first SCAN VIP to `node1`:

```
$ srvctl relocate scan -scannumber 1 -node node1
```

srvctl remove scan

Removes Oracle Clusterware resources from all SCAN VIPs.

Syntax

```
srvctl remove scan [-netnum network_number] [-force] [-noprompt]
```

Parameters

Table A-66 `srvctl remove scan` Command Parameters

Parameter	Description
<code>-netnum</code> <i>network_number</i>	The optional network number from which VIPs are obtained. If not specified, the VIPs are obtained from the same default network from which the <code>nodeapps</code> VIP is obtained.
<code>-force</code>	Removes the SCAN VIPs even though there are SCAN listeners running that are dependent on the SCAN VIPs.
<code>-noprompt</code>	Use this parameter to suppress all prompts.

Usage Notes

If you use the `-force` option, then SCAN VIPs that are running are not stopped before the dependent resources are removed, which may require manual cleanup.

Examples

An example of this command is:

```
$ srvctl remove scan -force
```

srvctl start scan

Starts all SCAN VIPs, by default, or a specific SCAN VIP, on all nodes or a specific node in the cluster.

Syntax

```
srvctl start scan [-scannumber ordinal_number] [-node node_name]
```

Parameters

Table A-67 srvctl start scan Command Parameters

Parameter	Description
<code>-scannumber <i>ordinal_number</i></code>	Optionally, you can specify an ordinal number that identifies which SCAN VIP you want to start. The range of values you can specify for this parameter is 1 to 3. If you do not use this parameter, then SRVCTL starts all the SCAN VIPs.
<code>-node <i>node_name</i></code>	Optionally, you can specify the name of a single node on which the SCAN VIP resides that you want to start. If you do not specify this parameter, then SRVCTL starts the SCAN VIPs on all nodes in the cluster.

Usage Notes

You can only use this command with Oracle Clusterware.

Example

The following example starts the SCAN VIP identified by the ordinal number 1 on the `crml` node:

```
$ srvctl start scan -scannumber 1 -node crml
```

srvctl status scan

Displays the status for all SCAN VIPs, by default, or a specific SCAN VIP.

Syntax

```
srvctl status scan [-scannumber ordinal_number] [-verbose]
```

Usage Notes

- You can only use this command with Oracle Clusterware.
- Optionally, you can specify an ordinal number that identifies a specific SCAN VIP for which you want to display the status. The range of values you can specify for this parameter is 1 to 3. If you do not use this parameter, then SRVCTL displays the status of all SCAN VIPs in the cluster.
- Optionally, you can use the `-verbose` parameter to display detailed output.

srvctl stop scan

Stops all SCAN VIPs, by default, that are running or in starting state, or stops a specific SCAN VIP identified by an ordinal number.

Syntax

```
srvctl stop scan [-scannumber ordinal_number] [-force]
```

Usage Notes

- You can only use this command with Oracle Clusterware.
- Optionally, you can specify an ordinal number that identifies which SCAN VIP you want to stop. The range of values you can specify for this parameter is 1 to 3. If you do not use this parameter, then SRVCTL stops all the SCAN VIPs.
- Optionally, you can use the `-force` parameter to stop the SCAN VIPs regardless of any dependencies.

Example

The following example stops the SCAN VIP identified by the ordinal number 1:

```
$ srvctl stop scan -scannumber 1
```

srvctl add scan_listener

Adds Oracle Clusterware resources for the SCAN listeners.

Syntax

```
srvctl add scan_listener [-listener lsnr_name_prefix] [-skip] [-netnum network_number]
[-endpoints "[TCP:]port_list[:FIREWALL={ON|OFF}][IPC:key][NMP:pipe_name]
[/{TCPS|SDP|EXADIRECT}port_list[:FIREWALL={ON|OFF}]]" [-invitednodes node_list] [-invitedsubnets subnet_list]
```

Parameters

Table A-68 srvctl add scan_listener Command Parameters

Parameter	Description
<code>-listener lsnr_name_prefix</code>	The SCAN listener name prefix.
<code>-skip</code>	Use this parameter to skip checking of the ports.
<code>-netnum network_number</code>	The optional network number from which SCAN VIPs are obtained. If you do not specify this parameter, then the SCAN VIPs are obtained from the same default network from which the <code>nodeapps</code> VIP is obtained.
<code>-endpoints "[TCP:]port_list[:FIREWALL={ON OFF}][IPC:key][NMP:pipe_name] [/{TCPS SDP EXADIRECT}port_list[:FIREWALL={ON OFF}]]"</code>	Protocol specifications for the SCAN listener. Use <code>port_list</code> to specify a comma-delimited list of TCP ports or SCAN listener endpoints. You can also specify endpoints for TCPS, SDP, and EXADIRECT ports. Note: You can modify this attribute using Online Resource Attribute Modification.

Table A-68 (Cont.) srvctl add scan_listener Command Parameters

Parameter	Description
<code>-invitednodes <i>node_list</i></code>	Use this parameter to specify a comma-delimited list of host names from outside the cluster that are allowed to register with the SCAN listener.
<code>-invitedsubnets <i>subnet_list</i></code>	Use this parameter to specify a comma-delimited list of subnets from outside the cluster that are allowed to register with the SCAN listener. You can specify the subnets using either CIDR notation or wildcards (such as 192.168.*).

Usage Notes

- The number of SCAN listener resources created is the same as the number of SCAN VIP resources.
- This command is only available with Oracle Clusterware.

Example

An example of this command is:

```
# srvctl add scan_listener -listener myscanlistener
```

srvctl config scan_listener

Displays the configuration information for all SCAN listeners, by default, or a specific listener identified by network number or ordinal_number.

Syntax

```
srvctl config scan_listener [[-netnum network_number] [-scannumber ordinal_number] | -all]
```

Parameters**Table A-69 srvctl config scan_listener Command Parameters**

Parameter	Description
<code>-netnum <i>network_number</i></code>	Use this parameter to view the configuration of the listener for a specific SCAN VIP.
<code>-scannumber <i>ordinal_number</i></code>	Use this parameter to specify any one of the three SCAN VIPs, using values from 1 to 3, for which you want to view the configuration of the listener.
<code>-all</code>	Alternative to specifying network or ordinal numbers, you can use this parameter to view the configuration of the listeners for all of the SCAN VIPs.

Usage Notes

This command is only available with Oracle Clusterware.

Example

This command returns output similar to the following:

```
$ srvctl config scan_listener -scannumber 1

SCAN Listener LISTENER_SCAN1 exists. Port: TCP:1529
Registration invited nodes:
Registration invited subnets:
SCAN Listener is enabled.
SCAN Listener is individually enabled on nodes:
SCAN Listener is individually disabled on nodes:
```

srvctl disable scan_listener

Disables all SCAN listeners, by default, or a specific listener identified by an ordinal number.

Syntax

```
srvctl disable scan_listener [-scannumber ordinal_number]
```

Usage Notes

- You can only use this command with Oracle Clusterware.
- Optionally, you can use the `-scannumber` parameter to specify any one of the three SCAN listeners. The range of values you can specify for this parameter is 1 to 3. If you do not use this parameter, then SRVCTL disables all SCAN listeners.

Example

The following example disables the SCAN listener identified as 1:

```
$ srvctl disable scan_listener -scannumber 1
```

srvctl enable scan_listener

Enables all SCAN listeners, by default, or a specific listener identified by its ordinal number.

Syntax

```
srvctl enable scan_listener [-scannumber ordinal_number]
```

Usage Notes

- This command is only available with Oracle Clusterware.
- Optionally, you can use the `-scannumber` parameter to specify any one of the three SCAN listeners you want to enable. The range of values you can specify for this parameter is 1 to 3. If you do not use this parameter, then SRVCTL disables all SCAN listeners.

Example

The following example enables the SCAN listener identified as 1:

```
$ srvctl enable scan_listener -scannumber 1
```

srvctl modify scan_listener

Modifies the SCAN listener to match that of the SCAN VIP, or modifies the SCAN listener endpoints or service registration restrictions.

Syntax

```
srvctl modify scan_listener {-update | -endpoints [TCP:]port_list[:FIREWALL={ON|OFF}]
[/IPC:key]
    [/NMP:pipe_name][/{TCPS|SDP|EXADIRECT}port_list[:FIREWALL={ON|OFF}]]"} [-
invitednodes node_list]
    [-invitedsubnets subnet_list]
```

Parameters

Table A-70 srvctl modify scan_listener Command Parameters

Parameter	Description
-update	Use this parameter to update SCAN listener configuration to match the current SCAN VIP configuration. This parameter adds new resources or removes existing SCAN listener resources to match the number of SCAN VIP resources.
-endpoints "[TCP:]port_list[:FIREWALL={ON OFF}][/ IPC:key] [/ NMP:pipe_name][/{TCPS SDP EXADIRECT}port_list[:FIREWALL={ON OFF}]]"	Protocol specifications for the SCAN listener. Use <i>port_list</i> to specify a comma-delimited list of TCP ports or listener endpoints. You can also specify endpoints for TCPS, SDP, and EXADIRECT ports.
-invitednodes <i>node_list</i>	Use this parameter to specify a comma-delimited list of host names from outside the cluster that are allowed to register with the SCAN listener.
-invitedsubnets <i>subnet_list</i>	Use this parameter to specify a comma-delimited list of subnets from outside the cluster that are allowed to register with the SCAN listener. You can specify the subnets using either CIDR notation or wildcards (such as 192.168.*).

Example

Assume your system currently has a SCAN named `scan_name1`, and you recently modified the DNS entry to resolve to three IP addresses instead of one. After running the `srvctl modify scan` command to create additional SCAN VIP resources, use the following command to create Oracle Clusterware resources for the additional two SCAN listeners to go with the two additional SCAN VIPs:

```
$ srvctl modify scan_listener -update
```

srvctl predict scan_listener

Predicts the consequences of SCAN listener failure.

Syntax

```
srvctl predict scan_listener -scannumber ordinal_number [-verbose]
```

Usage Notes

- Use the `-scannumber` parameter to specify any one of the three SCAN listeners for which you want to predict the consequences of a failure. The range of values you can specify for this parameter is 1 to 3.
- Optionally, you can use the `-verbose` parameter to display detailed output.

srvctl relocate scan_listener

Relocates a specific SCAN listener from its current hosting node to another node within the cluster.

Syntax

```
srvctl relocate scan_listener -scannumber ordinal_number [-node node_name]
```

Usage Notes

- You can only use this command with Oracle Clusterware.
- Specify an ordinal number that identifies which SCAN listener you want to relocate. The range of values you can specify for this parameter is 1 to 3.
- Optionally, you can specify the name of a single node to which you want to relocate the SCAN listener. If you do not specify this parameter, then SRVCTL chooses the node to which the SCAN listener is relocated.

Example

The following example relocates the SCAN listener identified as 3 to `node2` of the cluster:

```
$ srvctl relocate scan_listener -scannumber 3 -node node2
```

srvctl remove scan_listener

Removes Oracle Clusterware resources from all SCAN listeners.

Syntax

```
srvctl remove scan_listener [-netnum network_number] [-force] [-noprompt]
```

Parameters**Table A-71 srvctl remove scan_listener Command Parameters**

Parameter	Description
<code>-netnum network_number</code>	The optional network number from which SCAN VIPs are obtained. If you do not specify this parameter, then the SCAN VIPs are obtained from the same default network from which the <code>nodeapps</code> VIP is obtained.

Table A-71 (Cont.) srvctl remove scan_listener Command Parameters

Parameter	Description
-force	Removes the SCAN listener without stopping the SCAN listener if it is running.
-noprompt	Use this parameter to suppress all prompts.

Example

An example of this command is:

```
$ srvctl remove scan_listener -force
```

srvctl start scan_listener

Starts all SCAN listeners, by default, or a specific listener on all nodes or a specific node in the cluster.

Syntax

```
srvctl start scan_listener [-node node_name] [-scannumber ordinal_number]
```

Usage Notes

- You can only use this command with Oracle Clusterware.
- Optionally, you can use the `-node` parameter to specify the name of a single node on which you want to start a SCAN listener. If you do not use this parameter, then SRVCTL starts the SCAN listeners on all nodes in the cluster.
- Optionally, you can use the `-scannumber` parameter to specify an ordinal number that identifies which SCAN listener you want to start. The range of values you can specify for this parameter is 1 to 3. If you do not use this parameter, then SRVCTL starts all the SCAN listeners.

Example

The following example starts the SCAN listener identified as 1:

```
$ srvctl start scan_listener -scannumber 1
```

srvctl status scan_listener

Displays the status for all SCAN listeners, by default, or a specific listener.

Syntax

```
srvctl status scan_listener [-scannumber ordinal_number] [-verbose]
```

Usage Notes

- This command is only available with Oracle Clusterware.
- Optionally, you can use the `-scannumber` parameter to specify an ordinal number that identifies a specific SCAN listener. The range of values you can specify for

this parameter is 1 to 3. If you do not use this parameter, then the utility displays the status of all SCAN listeners in the cluster.

- Optionally, you can use the `-verbose` parameter to display detailed output.

srvctl stop scan_listener

Stops all SCAN listeners, by default, that are in a running or starting state, or a specific listener identified by an ordinal number.

Syntax

```
srvctl stop scan_listener [-scannumber ordinal_number] [-force]
```

Usage Notes

- This command is only available with Oracle Clusterware.
- Optionally, you can use the `-scannumber` parameter to specify an ordinal number that identifies which SCAN listener to stop. The range of values you can specify for this parameter is 1 to 3. If you do not specify this parameter, then SRVCTL stops all the SCAN listeners.
- Optionally, you can use the `-force` parameter to stop the SCAN listener regardless of any dependencies.

Example

The following example stops the SCAN listener identified as 1:

```
$ srvctl stop scan_listener -scannumber 1
```

srvctl update scan_listener

Updates the SCAN listeners to listen on the new endpoints.

Syntax

```
srvctl update scan_listener
```

Usage Notes

- You can only use this command with Oracle Clusterware.
- This command does not accept any additional parameters, except for `-help`.

srvctl relocate server

Relocates servers to a server pool in the cluster.

Syntax

```
srvctl relocate server -servers "server_name_list" -serverpool pool_name  
[-eval] [-force]
```

Parameters

Table A-72 `srvctl relocate server` Command Parameters

Parameter	Description
<code>-servers</code> <code>"server_name_list"</code>	Specify either a single server name or a comma-delimited list of server names enclosed in double quotation marks (") that you want to relocate to a different server pool.
<code>-serverpool pool_name</code>	Specify the name of the server pool to which you want to move servers.
<code>-eval</code>	Optionally, you can use this parameter to hypothetically evaluate the impact of the command on the system.
<code>-force</code>	Optionally, you can use this parameter to force the relocation of servers even if it means stopping some resources.

Example

The following example relocates two servers to a different server pool:

```
$ srvctl relocate server -servers "server1, server2" -serverpool sp3
```

srvctl status server

Displays the current state of specific servers.

Syntax

```
srvctl status server -server "server_name_list" [-detail]
```

Usage Notes

- Use the `-server` parameter to specify a single server name or a comma-delimited list of server names enclosed in double quotation marks (") for which you want to check the status.
- Optionally, you can use the `-detail` parameter to print detailed status information.

srvctl add service

Adds services to a database and assigns them to instances.

Syntax

Use this command with one of the following syntax models.

To add a service to a policy-managed database:

```
srvctl add service -db db_unique_name -service service_name [-eval]
  -serverpool server_pool [-pdb pluggable_database]
  [-cardinality {UNIFORM | SINGLETON}] [-edition edition_name]
  [-netnum network_number] [-role "[PRIMARY][,PHYSICAL_STANDBY][,LOGICAL_STANDBY]
  [,SNAPSHOT_STANDBY]"]
  [-policy {AUTOMATIC | MANUAL}] [-notification {TRUE | FALSE}] [-rfpool pool_name]
  [-clbgoal {SHORT | LONG}] [-failovertime {NONE|SESSION|SELECT|TRANSACTION}]
  [-rlbgoal {NONE | SERVICE_TIME | THROUGHPUT}] [-dtp {TRUE | FALSE}]
```

```
[-failovermethod {NONE | BASIC}] [-failoverretry failover_retries]
[-drain_timeout timeout] [-stopoption {NONE|IMMEDIATE|TRANSACTIONAL}]
[-failoverrestore {NONE|LEVEL1}] [-failoverdelay failover_delay]
[-gsmflags gsm_flags] [-sql_translation_profile sql_translation_profile]
[-global {TRUE | FALSE}] [-maxlag max_lag_time] [-commit_outcome {TRUE|FALSE}]
[-retention retention_time] [-replay_init_time replay_initiation_time]
[-session_state {STATIC | DYNAMIC}] [-pqservice pq_service]
[-pqpool pq_pool] [-css_critical {yes | no}] [-force]
```

To add a service to an administrator-managed database:

```
srvctl add service -database db_unique_name -service service_name
  [-pdb pluggable_database] [-eval]
  [-preferred preferred_list] [-available available_list]
  [-netnum network_number] [-tafpolicy {BASIC | NONE}]
  [-edition edition_name]
  [-role "[PRIMARY][,PHYSICAL_STANDBY][,LOGICAL_STANDBY][,SNAPSHOT_STANDBY]"
  [-policy {AUTOMATIC | MANUAL}] [-notification {TRUE | FALSE}]
  [-clbgoal {SHORT | LONG}] [-failovertime {NONE|SESSION|SELECT|TRANSACTION}]
  [-rlbgoal {NONE | SERVICE_TIME | THROUGHPUT}] [-dtp {TRUE | FALSE}]
  [-failovermethod {NONE | BASIC}] [-failoverretry failover_retries]
  [-drain_timeout timeout] [-stopoption {NONE|IMMEDIATE|TRANSACTIONAL}]
  [-failoverrestore {NONE|LEVEL1}] [-failoverdelay failover_delay]
  [-sql_translation_profile sql_translation_profile]
  [-global {TRUE | FALSE}] [-maxlag max_lag_time] [-commit_outcome {TRUE|FALSE}]
  [-retention retention_time] [-replay_init_time replay_initiation_time]
  [-session_state {STATIC|DYNAMIC}] [-force] [-verbose]
```

To update the preferred and available lists of an existing service:

```
srvctl add service -db db_unique_name -service service_name
  -update {-preferred preferred_list | -available available_list} [-force]
  [-verbose]
```

Parameters

The following table lists and describes all the `srvctl add service` parameters and whether they can be used when adding a service to either an Oracle RAC database or noncluster database.

Table A-73 `srvctl add service` Command Parameters

Parameter	Description
<code>-db db_unique_name</code>	Unique name for the database.
<code>-service service_name</code>	The <code>service_name.service_domain</code> should be unique within the cluster unless you want to spread connections across multiple databases that offer the same service. If you do not specify the service domain as part of the service name (such as <code>sales.example.com</code>), then the <code>DB_DOMAIN</code> database attribute is appended to the service name. Note: The <code>-service</code> parameter has a 4 kilobyte (KB) limit for its value. Therefore, the total length of the names of all services assigned to an instance cannot exceed 4 KB.
<code>-eval</code>	Use this parameter to hypothetically evaluate the impact of the command on the system. Note: You can only use this parameter with a policy-managed service.

Table A-73 (Cont.) srvctl add service Command Parameters

Parameter	Description
<code>-edition <i>edition_name</i></code>	<p>The initial session edition of the service.</p> <p>When an edition is specified for a service, all subsequent connections that specify the service use this edition as the initial session edition. However, if a session connection specifies a different edition, then the edition specified in the session connection is used for the initial session edition.</p> <p>SRVCTL does not validate the specified edition name. During connection, the connect user must have <code>USE</code> privilege on the specified edition. If the edition does not exist or if the connect user does not have <code>USE</code> privilege on the specified edition, then an error is raised.</p>
<code>-sql_translation_profile <i>profile_name</i></code>	<p>Use this parameter to specify a SQL translation profile for a service that you are adding after you have migrated applications from a non-Oracle database to an Oracle database.</p> <p>This parameter corresponds to the SQL translation profile parameter in the <code>DBMS_SERVICE</code> service attribute.</p> <p>Notes:</p> <ul style="list-style-type: none"> • Before using the SQL translation feature, you must migrate all server-side application objects and data to the Oracle database. • Use the <code>srvctl config service</code> command to display the SQL translation profile.
<code>-preferred <i>preferred_list</i></code>	<p>A list of preferred instances on which the service runs when the database is administrator managed.</p> <p>The list of preferred instances must be mutually exclusive with the list of available instances.</p> <p>Note: This parameter can be used only with Oracle RAC and only for administrator-managed databases.</p>
<code>-available <i>available_list</i></code>	<p>A list of available instances to which the service fails over when the database is administrator managed.</p> <p>The list of available instances must be mutually exclusive with the list of preferred instances.</p> <p>Note: This parameter can be used only with Oracle RAC and only for administrator-managed databases.</p>
<code>-serverpool <i>server_pool</i></code>	<p>The name of a server pool used when the database is policy managed.</p> <p>Note: This parameter can be used only with Oracle RAC and only for policy-managed databases.</p>
<code>-cardinality {UNIFORM SINGLETON}</code>	<p>The cardinality of the service, either <code>UNIFORM</code> (offered on all instances in the server pool) or <code>SINGLETON</code> (runs on only one instance at a time).</p> <p>Notes:</p> <ul style="list-style-type: none"> • This parameter can be used only with Oracle RAC and only for policy-managed databases. • For policy-managed Oracle RAC One Node databases, all services must be <code>SINGLETON</code>.

Table A-73 (Cont.) `srvctl add service` Command Parameters

Parameter	Description
<code>-netnum network_number</code>	Use this parameter to determine on which network this service is offered. The service is configured to depend on VIPs from the specified network. Notes: <ul style="list-style-type: none"> If you omit this parameter, then the default is taken from the database configuration, which you specify using <code>srvctl add database</code> or <code>srvctl modify database</code>, with the <code>-defaultnetwork</code> parameter to specify the default network for that database's services. This parameter can be used only with Oracle RAC and Oracle RAC One Node database configurations.
<code>-tafpolicy {BASIC NONE}</code>	TAF policy specification (for administrator-managed databases only).
<code>-role "[PRIMARY] [, PHYSICAL_STANDBY] [, LOGICAL_STANDBY] [, SNAPSHOT_STANDBY]"</code>	The service role. You can specify one or more roles in a comma-delimited list. Use this option to indicate that the service should only be automatically started upon database open when the Oracle Data Guard database role matches one of the specified service roles. Using SRVCTL to manually start a service is not affected by the service role. Note: The <code>-role</code> parameter is only used at database startup and by the Oracle Data Guard Broker. All manual service startup must specify the name of the service to be started by the user.
<code>-policy {AUTOMATIC MANUAL}</code>	Service management policy. If <code>AUTOMATIC</code> (the default), then the service is automatically started upon restart of the database, either by a planned restart (with SRVCTL) or after a failure. Automatic restart is also subject to the service role, however (the <code>-role</code> parameter). If <code>MANUAL</code> , then the service is never automatically restarted upon planned restart of the database (with SRVCTL). A <code>MANUAL</code> setting does not prevent Oracle Clusterware from monitoring the service when it is running and restarting it if a failure occurs. Note: Using CRSCCTL to stop and start the Oracle Clusterware restarts the service in the same way that a failure does.
<code>-notification {TRUE FALSE}</code>	Enable Fast Application Notification (FAN) for OCI connections.
<code>-rfpool pool_name</code>	Specify the name of the reader farm server pool.
<code>-dtp {TRUE FALSE}</code>	Indicates whether Distributed Transaction Processing should be enabled for this service. This service will either be a singleton service in a policy-managed database or a preferred service on a single node in an administrator-managed database. Note: This parameter can be used only with Oracle RAC.
<code>-clbgoal {SHORT LONG}</code>	Connection Load Balancing Goal. Use a value of <code>SHORT</code> for this parameter for run-time load balancing, or if using an integrated connection pool. Use a value of <code>LONG</code> for this parameter for long running connections, such as batch jobs, that you want balanced by the number of sessions per node for the service.

Table A-73 (Cont.) srvctl add service Command Parameters

Parameter	Description
-rlbgoal {NONE SERVICE_TIME THROUGHPUT}	Runtime Load Balancing Goal (for the Load Balancing Advisory). Set this parameter to SERVICE_TIME to balance connections by response time. Set this parameter to THROUGHPUT to balance connections by throughput.
-failovertype {NONE SESSION SELECT TRANSACTION}	Failover type. To enable Application Continuity for Java, set this parameter to TRANSACTION. To enable TAF for OCI, set this parameter to SELECT or SESSION. Note: If you set -failovertype to TRANSACTION, then you must set -commit_outcome to TRUE.
-failovermethod {NONE BASIC}	TAF failover method (for backward compatibility only). If the failover type (-failovertype) is set to a value other than NONE, then you should choose BASIC for this parameter. Note: This parameter can be used only with Oracle RAC.
-failoverretry failover_retries	For Application Continuity and TAF, this parameter determines the number of attempts to connect after an incident.
-drain_timeout timeout	Specify the time, in seconds, allowed for resource draining to be completed. Accepted values are an empty string (""), 0, or any positive integer. The default value is an empty string, which means that this parameter is not set. If it is set to 0, then draining occurs, immediately. The draining period is intended for planned maintenance operations. During the draining period, all current client requests are processed, but new requests are not accepted. When set on the service this value is used when the command line value is not set.
-stopoption {NONE IMMEDIATE TRANSACTIONAL}	Specify the mode in which the service is stopped. When set on the service, this value is used if you do not set the value on the command line. IMMEDIATE permits sessions to drain before the service is stopped. TRANSACTIONAL permits sessions to drain for the amount of time specified in the -drain_timeout parameter. The service is stopped when the time limit is reached and any remaining sessions are terminated. If you specify NONE, then no sessions are terminated. The default is taken from the service setting, when specified on the service. Otherwise the default is NONE. Note: You must use the -stopoption parameter with the -force parameter.
-failoverrestore {NONE LEVEL1}	For Application Continuity, when you set the -failoverrestore parameter, session states are restored before replaying. Use LEVEL1 for ODP.NET and Java with Application Continuity to restore the initial state. For OCI applications using TAF or Application Continuity, setting -failoverrestore to LEVEL1 restores the current state. If the current state differs from the initial state, then a TAF callback is required. This restriction applies only to OCI.

Table A-73 (Cont.) `srvctl add service` Command Parameters

Parameter	Description
<code>-failoverdelay</code> <code>failover_delay</code>	For Application Continuity and TAF, this parameter specifies the time delay (in seconds) between reconnect attempts per incident at failover.
<code>-gsmflags</code> <code>gsm_flags</code>	Set locality and region failover values.
<code>-pdb</code> <code>pluggable_database</code>	The name of the pluggable database (PDB). Note: You can specify a PDB property when you create or modify a service. The PDB property associates the service with the specified PDB. You can view the PDB property for a service by querying the <code>ALL_SERVICES</code> data dictionary view or, when using the <code>SRVCTL</code> utility, by running the <code>srvctl config service</code> command. When you create or modify a service with the specified PDB, <code>SRVCTL</code> does not check if the PDB exists. Before running this command, you must ensure that the PDB exists.
<code>-global</code> { <code>TRUE</code> <code>FALSE</code> }	Indicates whether this is a Global Data Services service. Note: This parameter can only be used with Global Data Services.
<code>-maxlag</code> <code>maximum_lag_time</code>	Maximum replication lag time in seconds. Must be a non-negative integer. The default value is <code>ANY</code> .
<code>-commit_outcome</code> { <code>TRUE</code> <code>FALSE</code> }	Enable Transaction Guard; when set to <code>TRUE</code> , the commit outcome for a transaction is accessible after the transaction's session fails due to a recoverable outage.
<code>-retention</code> <code>retention_time</code>	If <code>-commit_outcome</code> is set to <code>TRUE</code> , then this parameter determines the amount of time (in seconds) that the commit outcome is retained in the database.
<code>-replay_init_time</code> <code>replay_initialization_time</code>	For Application Continuity, this parameter specifies the difference between the time, in seconds, of original execution of the first operation of a request and the time that the replay is ready to start after a successful reconnect. Application Continuity will not replay after the specified amount of time has passed. This parameter is intended to avoid the unintentional execution of a transaction when a system is recovered after a long period. The default is 5 minutes (300). The maximum value is 24 hours (86400). If the <code>-failover_type</code> parameter is not set to <code>TRANSACTION</code> , then you cannot use this parameter.
<code>-session_state</code> { <code>STATIC</code> <code>DYNAMIC</code> }	For Application Continuity, this parameter specifies whether the session state that is not transactional is changed by the application. Oracle recommends a setting of <code>DYNAMIC</code> for most applications. Note: This parameter is considered only if <code>-failovertime</code> is set to <code>TRANSACTION</code> for Application Continuity. It describes how non-transactional is changed during a request. Examples of session state are NLS settings, optimizer preferences, event settings, PL/SQL global variables, temporary tables, advanced queues, LOBs, and result cache. If non-transactional values change after the request starts, then use the default, <code>DYNAMIC</code> . Most applications should use <code>DYNAMIC</code> mode. If you are unsure, then use <code>DYNAMIC</code> mode.
<code>-pqs-service</code> <code>pq_service</code>	Specify a parallel query service name.
<code>-pqpool</code> <code>pq_pool</code>	Specify a parallel query server pool.

Table A-73 (Cont.) srvctl add service Command Parameters

Parameter	Description
<code>-update</code> <code>{-preferred</code> <code>new_preferred_instance</code> <code> -available</code> <code>new_available_instance</code> <code>}</code>	Add a new preferred or available instance to the service configuration. <code>-preferred</code> specifies the name of the instance to add to the list of preferred instances for the service. <code>-available</code> specifies the name of the instance to add to the list of available instances for the service.
<code>-css_critical</code> <code>{yes </code> <code>no}</code>	You can add weight to a service by setting this parameter to <code>YES</code> . In the event of a node failure within the cluster, Oracle Clusterware will evict the node with the least amount of weight, ensuring that critical services remain available. Note: You can use this parameter only on an administrator-managed node. Should the node become policy managed, at some point, this parameter will no longer apply.
<code>-verbose</code>	Display verbose output.
<code>-force</code>	Force the add operation even though a listener is not configured for a network.

Usage Notes

This command does not accept placement parameters for Oracle RAC One Node databases.

Examples

Use this example syntax to add the `gl.example.com` service to the `my_rac` database with Fast Application Notification enabled for OCI connections, a failover method of `BASIC`, a Connection Load Balancing Goal of `LONG`, a failover type of `SELECT`, and 180 failover retries with a failover delay of 5 seconds:

```
$ srvctl add service -db my_rac -service gl.example.com -notification TRUE -
failovermethod BASIC -failovertime SELECT -failoverretry 180 -failoverdelay 5 -
clbgoal LONG
```

Use this example syntax to add a named service to a database with preferred instances and available instances and enabled for TAF:

```
$ srvctl add service -db crm -service sales -preferred crm01,crm02 -available crm03 -
tafpolicy BASIC
```

srvctl config service

Displays the configuration for a service.

Syntax

```
srvctl config service {-db db_unique_name [-service service_name]  
| -serverpool pool_name [-db db_unique_name]} [-verbose]
```

Parameters

Table A-74 `srvctl config service` Command Parameters

Parameter	Description
<code>-db db_unique_name</code>	Unique name for the database.
<code>-service service_name</code>	Optionally, you can specify the name of a service. If you do not use this parameter, then SRVCTL displays the configuration information for all services configured for the database.
<code>-serverpool pool_name</code>	Alternatively, you can use this parameter to specify the name of a server pool for which you want to view the service configuration. Optionally, you can also specify a particular database on which the server pool resides.
<code>-verbose</code>	Displays verbose output.

Usage Notes

The `srvctl config service` command shows exactly the string value you specified for the edition using the `srvctl add | modify service` commands. If you specified the edition in upper case, then `srvctl config service` displays upper case. If it is surrounded by double quotation marks ("`"`"), then the command displays the double quotation marks. Otherwise, the command displays an empty string.

Examples

This command returns information similar to the following for a policy-managed database:

```
$ srvctl config service -db crm -service webapps
```

```
Service name: webapps
Service is enabled
Server pool: sales
Cardinality: SINGLETON
Disconnect: false
Service role: PRIMARY
Management policy: AUTOMATIC
DTP transaction: false
AQ HA notifications: false
Failover type: NONE
Failover method: NONE
TAF failover retries: 0
TAF failover delay: 0
Connection Load Balancing Goal: LONG
Runtime Load Balancing Goal: NONE
TAF policy specification: NONE
Service is enabled on nodes:
Service is disabled on nodes:
Edition: "my Edition"
```

This command returns information similar to the following for an administrator-managed database:

```
$ srvctl config service -db crm -service webapps
```

```
Service name: webapps
Service is enabled
```

```

Server pool: sales
Cardinality: 1
Disconnect: false
Service role: PRIMARY
Management policy: AUTOMATIC
DTP transaction: false
AQ HA notifications: false
Failover type: NONE
Failover method: NONE
TAF failover retries: 0
TAF failover delay: 0
Connection Load Balancing Goal: LONG
Runtime Load Balancing Goal: NONE
TAF policy specification: NONE
Preferred instances: crm_1
Available instances:
Edition: "my Edition"

```

Service configuration for administrator-managed Oracle RAC One Node databases displays the one instance as preferred.

srvctl disable service

Disables a service.

Disabling an entire service affects all of the instances, disabling each one. If the entire service is already disabled, then running this command on the entire service 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

```

srvctl disable service -db db_unique_name -services "service_name_list"
    [-instance instance_name | -node node_name]

```

Parameters

Table A-75 srvctl disable service Command Parameters

Parameter	Description
-db <i>db_unique_name</i>	Specify a unique name for the database for which you want to disable the service.
-services " <i>service_name_list</i> "	Specify a comma-delimited list of service names enclosed in double quotation marks (" "), or a single service name, that you want to disable.
-instance <i>instance_name</i>	Optionally, you can specify the name of the instance for which you want to disable the service.
	Notes: <ul style="list-style-type: none"> Use this parameter with administrator-managed databases. You can only use this parameter with Oracle Clusterware and Oracle RAC.

Table A-75 (Cont.) srvctl disable service Command Parameters

Parameter	Description
<code>-node <i>node_name</i></code>	<p>Alternative to using the <code>-instance</code> parameter, you can use this parameter to specify the name of the node on which to disable the service.</p> <p>Notes:</p> <ul style="list-style-type: none"> • Use this parameter with policy-managed databases. • You can only use this parameter with Oracle Clusterware and Oracle RAC.

Examples

The following example globally disables two services for the CRM database:

```
$ srvctl disable service -db crm -service "crm,marketing"
```

The following example disables a service for the CRM database that is running on the CRM1 instance, resulting in the service still being available for the database, but on one less instance:

```
$ srvctl disable service -db crm -service crm -instance crm1
```

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 on each one. If the entire service is already enabled, then running this command does not affect all of the instances and enable them but, instead, returns an error. Therefore, you cannot always use the entire set of service operations to manipulate the service indicators for each instance.

Syntax

```
srvctl enable service -db db_unique_name -service "service_name_list"
  [-instance instance_name | -node node_name]
```

Parameters

Table A-76 srvctl enable service Command Parameters

Parameter	Description
<code>-db <i>db_unique_name</i></code>	Specify a unique name for the database for which you want to enable the service.
<code>-service "<i>service_name_list</i>"</code>	Specify a single service name or a comma-delimited list of service names enclosed in double quotation marks (") that you want to enable.
<code>-instance <i>instance_name</i></code>	<p>Optionally, you can use this parameter to specify the name of the database instance where you want the service to run.</p> <p>Notes:</p> <ul style="list-style-type: none"> • Use this parameter for administrator-managed databases. • You can only use this parameter with Oracle Clusterware and Oracle RAC.

Table A-76 (Cont.) srvctl enable service Command Parameters

Parameter	Description
<code>-node <i>node_name</i></code>	<p>Alternative to using the <code>-instance</code> parameter, you can use this parameter to specify the name of the node on which to enable the service.</p> <p>Notes:</p> <ul style="list-style-type: none"> • Use this parameter with policy-managed databases. • You can only use this parameter with Oracle Clusterware and Oracle RAC.

Examples

The following example globally enables a service:

```
$ srvctl enable service -db crm -service crm
```

The following example enables a service to use a preferred instance:

```
$srvctl enable service -db crm -service crm -instance crm1
```

srvctl modify service

Modifies service configuration.

This command supports some online modifications to the service, such as:

- Moving a service member from one instance to another
- Performing online changes to service attributes from `DBMS_SERVICE` (for example, failover delay, runtime load balancing goal, and so on)
- Adding a new preferred or available instance
- Removing preferred or available instances for a service

▲ Caution:

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 Parameters

Use one of the following forms of the `srvctl modify service` command, depending on the task you want to perform, with the specified syntax:

To move a service from one instance to another:

```
srvctl modify service -db db_unique_name -service service_name
    -oldinst old_instance_name -newinst new_instance_name [-force]
```

 **Note:**

This form of the command is only available with Oracle Clusterware.

Table A-77 `srvctl modify service` Parameters for Moving a Service

Parameter	Description
<code>-db db_unique_name</code>	Specify the unique name for the database.
<code>-service service_name</code>	Specify a service name. If you do not specify a service name, then SRVCTL moves all services.
<code>-oldinst old_instance_name</code>	Specify the name of the instance from which you want to move the service.
<code>-newinst new_instance_name</code>	Specify the name of the instance to which you want to move the service.
<code>-force</code>	Disconnects all sessions during stop or relocate service operations.

To change an available instance to a preferred instance for a service:

```
srvctl modify service -db db_unique_name -service service_name
  -available avail_inst_name -toprefer [-force]
```

 **Note:**

This form of the command is only available with Oracle Clusterware and does not accept placement parameters for Oracle RAC One Node databases.

Table A-78 `srvctl modify service` Parameters for Changing to a Preferred Instance

Parameter	Description
<code>-db db_unique_name</code>	Specify the unique name for the database.
<code>-service service_name</code>	Specify the name of the service you want to modify.
<code>-available available_inst_name</code>	Specify the name of the available instance you want to change.
<code>-toprefer</code>	Specify this parameter to change the instance status to preferred.
<code>-force</code>	Disconnect all sessions during stop or relocate service operations. For planned operations, the user experience is best if using an Oracle Connection Pool with FAN, and not forcing disconnect. The FAN planned event causes the Oracle pool to drain the requests with no interruption to the users.

To change the available and preferred status for multiple instances:

```
srvctl modify service -db db_unique_name -service service_name
  -modifyconfig -preferred "preferred_list" [-available "available_list"]
  [-force]
```



Note:

This form of the command is only available with Oracle Clusterware and does not accept placement parameters for Oracle RAC One Node databases.

Table A-79 `srvctl modify service` Parameters for Changing Status of Multiple Instances

Parameter	Description
<code>-db db_unique_name</code>	Specify the unique name for the database.
<code>-service service_name</code>	Specify the name of the service you want to modify.
<code>-modifyconfig</code>	This parameter directs SRVCTL to use only the instances named for this service (unnamed instances already assigned to the service are removed).
<code>-preferred</code> <code>"preferred_instance_list"</code>	Specify a comma-delimited list of preferred instances enclosed within double quotation marks ("").
<code>-available</code> <code>"available_instance_list"</code>	Specify a comma-delimited list of available instances enclosed within double quotation marks ("").
<code>-force</code>	Disconnect all sessions during stop or relocate service operations. For planned operations, the user experience is best if using an Oracle Connection Pool with FAN, and not forcing disconnect. The FAN planned event causes the connection pool to drain the requests with no interruption to the users.

To modify other service attributes or to modify a service for Oracle Clusterware:

```

srvctl modify service -db db_unique_name -service service_name [-eval]
[-serverpool pool_name] [-cardinality {UNIFORM|SINGLETON}]
[-drain_timeout timeout] [-stopoption {NONE|IMMEDIATE|TRANSACTIONAL}]
[-pqservic pgsvc_name] [-pqpool pq_pool_list]
[-pdb pluggable_database] [-tafpolicy {BASIC|NONE}]
[-edition edition_name] [-role "{PRIMARY}|{PHYSICAL_STANDBY}|
{LOGICAL_STANDBY}|{SNAPSHOT_STANDBY}"] [-notification {TRUE|FALSE}]
[-dtp {TRUE|FALSE}] [-clbgoal {SHORT|LONG}] [-rlbgoal {NONE|SERVICE_TIME|
THROUGHPUT}]
[-failovertime {NONE|SESSION|SELECT|TRANSACTION}] [-failovermethod {NONE|BASIC}]
[-failoverrestore {NONE|LEVEL1}] [-failoverretry failover_retries]
[-failoverdelay failover_delay] [-policy {AUTOMATIC|MANUAL}]
[-sql_translation_profile profile_name] [-commit_outcome {TRUE|FALSE}]
[-retention retention_time] [-replay_init_time replay_initiation_time]
[-session_state {STATIC|DYNAMIC}] [-global_override] [-verbose] [-force]

```

Table A-80 `srvctl modify service` Parameters

Parameter	Description
<code>-db db_unique_name</code>	Specify the unique name for the database.
<code>-service service_name</code>	Specify the name of the service you want to modify.

Table A-80 (Cont.) `srvctl modify service` Parameters

Parameter	Description
<code>-eval</code>	Use this parameter to hypothetically evaluate the impact of the command on the system. Note: You can only use this parameter with a policy-managed service.
<code>-serverpool <i>pool_name</i></code>	The name of a server pool used when the database is policy managed. Note: This parameter can be used only with Oracle RAC and only for policy-managed databases.
<code>-cardinality {UNIFORM SINGLETON}</code>	Specify the cardinality of the service, either <code>UNIFORM</code> (offered on all instances in the server pool) or <code>SINGLETON</code> (runs on only one instance at a time). Note: This parameter can be used only with Oracle Clusterware.
<code>-drain_timeout <i>timeout</i></code>	Specify the time, in seconds, allowed for resource draining to be completed. Accepted values are an empty string (""), 0, or any positive integer. The default value is an empty string, which means that this parameter is not set. If it is set to 0, then draining occurs, immediately. The draining period is intended for planned maintenance operations. During the draining period, all current client requests are processed, but new requests are not accepted. When set on the service this value is used when the command line value is not set.
<code>-stopoption {NONE IMMEDIATE TRANSACTIONAL}</code>	Specify the mode in which the service is stopped. If you set this parameter on the service, then this value is used if you do not set the value on the command line. <code>IMMEDIATE</code> permits sessions to drain before the service is stopped. <code>TRANSACTIONAL</code> permits sessions to drain for the amount of time specified in the <code>-drain_timeout</code> parameter. The service is stopped when the time limit is reached and any remaining sessions are terminated. If you specify <code>NONE</code> , then no sessions are terminated. The default is taken from the service setting, when specified on the service. Otherwise the default is <code>NONE</code> . Note: You must use the <code>-stopoption</code> parameter with the <code>-force</code> parameter.
<code>-pqserv <i>pqsvc_name</i></code>	Specify a comma-delimited list of parallel query service names.
<code>-pqpool <i>pq_pool_list</i></code>	Specify a comma-delimited list of parallel query server pool names.
<code>-pdb <i>pluggable_database</i></code>	Specify the name of a pluggable database (PDB). Note: You can specify a PDB property when you create or modify a service. The PDB property associates the service with the specified PDB. You can view the PDB property for a service by querying the <code>ALL_SERVICES</code> data dictionary view or, when using the <code>SRVCTL</code> utility, by running the <code>srvctl config service</code> command. When create or modify a service with the specified PDB, <code>SRVCTL</code> does not check if the PDB exists. Before running this command, you must ensure that the PDB exists.
<code>-tafpolicy {BASIC NONE}</code>	Specify the transaction failover (TAF) policy (for administrator-managed databases only).

Table A-80 (Cont.) srvctl modify service Parameters

Parameter	Description
-edition <i>edition_name</i>	<p>The initial session edition of the service.</p> <p>When an edition is specified for a service, all subsequent connections that specify the service use this edition as the initial session edition. However, if a session connection specifies a different edition, then the edition specified in the session connection is used for the initial session edition.</p> <p>SRVCTL does not validate the specified edition name. During connection, the connect user must have <code>USE</code> privilege on the specified edition. If the edition does not exist or if the connect user does not have <code>USE</code> privilege on the specified edition, then an error is raised.</p>
-role "[PRIMARY] [,PHYSICAL_STANDBY] [,LOGICAL_STANDBY] [,SNAPSHOT_STANDBY]"	<p>The database modes for which the service should be started automatically. You can specify one or more roles in a comma-delimited list.</p> <p>Note: The <code>-role</code> parameter is only used at database startup and by the Oracle Data Guard Broker. All manual service startup must specify the name of the service to be started by the user.</p>
-notification {TRUE FALSE}	Specify a value of <code>TRUE</code> to enable Fast Application Notification (FAN) for Oracle Call Interface (OCI) connections.
-dtp {TRUE FALSE}	Specify <code>TRUE</code> to enable Distributed Transaction Processing for this service. This ensures that the service is offered at exactly one instance at a time for XA affinity.
-clbgoal {SHORT LONG}	Use this parameter to set a connection load balancing goal: set to <code>SHORT</code> if using runtime load balancing, set to <code>LONG</code> for long running connections, such as batch jobs or older SQL*Forms style.
-rlbgoal {NONE SERVICE_TIME THROUGHPUT}	Use this parameter to set a runtime load balancing goal. Set this parameter to <code>SERVICE_TIME</code> to balance connections by response time. Set this parameter to <code>THROUGHPUT</code> to balance connections by throughput.
-failovertype {NONE SESSION SELECT TRANSACTION}	<p>Use this parameter to set the failover type.</p> <p>To enable Application Continuity, set this parameter to <code>TRANSACTION</code>. To enable TAF, set this parameter to <code>SELECT</code> or <code>SESSION</code>.</p>
-failovermethod {NONE BASIC}	Specify the TAF failover method (for backward compatibility only).
-failoverrestore [NONE LEVEL1]	<p>For Application Continuity, when you set the <code>-failoverrestore</code> parameter, session states are restored before replaying. Use <code>LEVEL1</code> for ODP.NET and Java with Application Continuity to restore the initial state.</p> <p>For OCI applications using TAF or Application Continuity, setting <code>-failoverrestore</code> to <code>LEVEL1</code> restores the current state. If the current state differs from the initial state, then a TAF callback is required. This restriction applies only to OCI.</p>
-failoverretry <i>failover_retries</i>	For Application Continuity and TAF, specify the number of attempts to connect after an incident.
-failoverdelay <i>failover_delay</i>	For Application Continuity and TAF, specify the time delay (in seconds) between reconnect attempts per incident at failover.
-policy {AUTOMATIC MANUAL}	Specify the service management policy.

Table A-80 (Cont.) `srvctl modify service` Parameters

Parameter	Description
- <code>sql_translation_profile profile_name</code>	Use this parameter to specify a SQL translation profile for a service that you are modifying after you have migrated applications from a non-Oracle database to an Oracle database. If you want to set the SQL translation profile to a NULL value, then you must enter an empty string after the <code>-p</code> flag. Note: Before using the SQL translation feature, you must migrate all server-side application objects and data to the Oracle database.
<code>-commit_outcome {TRUE FALSE}</code>	Enable Transaction Guard; when set to <code>TRUE</code> , the commit outcome for a transaction is accessible after the transaction's session fails due to a recoverable outage.
<code>-retention retention_time</code>	For Transaction Guard (with the <code>-commit_outcome</code> parameter set to <code>TRUE</code>); this parameter determines the amount of time (in seconds) that the commit outcome is retained in the database.
<code>-replay_init_time replay_initiation_time</code>	For Application Continuity; this parameter specifies the time, in seconds, from when the original request started. Application Continuity will not replay after the specified amount of time has passed. This attribute avoids the unintentional replay of a request when a system is recovered after a long period. The default value is 300 (5 minutes).
<code>-session_state {STATIC DYNAMIC}</code>	For Application Continuity; this parameter specifies whether the session state that is not transactional is changed by the application. Oracle recommends a value of <code>DYNAMIC</code> for most applications. If you are unsure which value to use or you can customize the application, then use <code>DYNAMIC</code> . <code>STATIC</code> mode is on request for database-agnostic applications, that never change non-transactional state inside a request.
<code>-global_override</code>	Override value to modify the global service attributes. Use this parameter with the <code>-role</code> , <code>-policy</code> , <code>-notification</code> , <code>-failovertime</code> , <code>-failovermethod</code> , <code>-failoverdelay</code> , <code>-failoverretry</code> and <code>-edition</code> parameters.
<code>-verbose</code>	Display verbose output.
<code>-force</code>	Force the modify operation, stopping the service on some nodes as necessary.

Usage Notes

- When performing online changes to service attributes (for example, failover delay, Runtime Load Balancing Goal, and so on), the changes take effect only when the service is next (re)started.
- 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.
- 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.

Because 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.

Examples

An example of moving a service member from one instance to another is:

```
$ srvctl modify service -db crm -service crm -oldinst crm1 -newinst crm2
```

An example of changing an available instance to a preferred instance is:

```
$ srvctl modify service -db crm -service crm -available crm1 -toprefer
```

The following command exchanges a preferred and available instance:

```
$ srvctl modify service -db crm -service crm -modifyconfig -preferred "crm1" \
-available "crm2"
```

srvctl predict service

Predicts the consequences of service failure.

Syntax

```
srvctl predict service -db db_unique_name -service service_name [-verbose]
```

Parameters

Table A-81 srvctl predict service Command Parameters

Parameter	Description
-db <i>db_unique_name</i>	Specify the unique name for the database on which the service operates that you want to check.
-service <i>service_name</i>	Specify a single service name or a comma-delimited list of service names enclosed in double quotation marks (" ") that you want to check.
-verbose	Optionally, you can use this parameter to display detailed output.

srvctl relocate service

Temporarily relocates the specified service names from one specified instance to another specified instance.

This command works on only one source instance and one target instance at a time, relocating a service or all services from a single source instance to a single target instance.

Syntax

```

srvctl relocate service -db db_unique_name [-service service_name
| -pdb pluggable_database] {-oldinst old_inst_name
[-newinst new_inst_name] | -currentnode source_node [-targetnode target_node]} [-
drain_timeout timeout]
[-wait YES | NO] [-pg] [-force [-noreplay]] [-eval] [-verbose]

```

Parameters

Table A-82 srvctl relocate service Command Parameters

Parameter	Description
-db <i>db_unique_name</i>	Specify a unique name for the database on which the service operates that you want to relocate.
-service <i>service_name</i>	Specify the name of the service you want to relocate. If you do not specify any services, then all services that can be relocated, are relocated. Those that cannot be relocated remain in place.
-pdb <i>pluggable_database</i>	Use this parameter to relocate services running on a specific pluggable database.
-oldinst <i>old_inst_name</i>	Specify the name of the instance from which you are relocating the service.
-newinst <i>new_inst_name</i>	Specify the name of the instance to which you are relocating the service. This parameter is optional. If you do not specify an instance, then Oracle Clusterware chooses a new one. Note: If you are using an administrator-managed database, then you must use the -oldinst and -newinst parameters and the target instance must be on the preferred or available list for the service.
-currentnode <i>source_node</i>	Name of the node where the service is currently running.
-targetnode <i>target_node</i>	Name of node where the service is to be relocated. If you do not specify a target node, then Oracle Clusterware chooses a new location. Note: If you are using a policy-managed database, then you must use the -currentnode and -targetnode parameters.
-drain_timeout <i>timeout</i>	Specify the time, in seconds, allowed for resource draining to be completed. Accepted values are an empty string (""), 0, or any positive integer. The default value is an empty string, which means that this parameter is not set. If it is set to 0, then draining occurs, immediately. The draining period is intended for planned maintenance operations. During the draining period, all current client requests are processed, but new requests are not accepted. When set on the service this value is used when the command line value is not set.
-wait YES NO	Choose YES to wait until service draining is completed on the node from which you are relocating the service.

Table A-82 (Cont.) srvctl relocate service Command Parameters

Parameter	Description
<code>-stopoption option</code>	Specify the mode in which the service is stopped. When set on the service, this value is used if you do not set the value on the command line. IMMEDIATE permits sessions to drain before the service is stopped. TRANSACTIONAL permits sessions to drain for the amount of time specified in the <code>-drain_timeout</code> parameter. The service is stopped when the time limit is reached and any remaining sessions are terminated. If you specify NONE, then no sessions are terminated. The default is taken from the service setting, when specified on the service. Otherwise the default is NONE. Note: You must use the <code>-stopoption</code> parameter with the <code>-force</code> parameter.
<code>-pq</code>	Performs the action on a parallel query service.
<code>-force [-noreplay]</code>	Disconnect all sessions during stop or relocate service operations. Optionally, you can specify the <code>-noreplay</code> parameter if you do not want Application Continuity to replay in-flight transactions after a session is terminated during stop or relocate service operations. The <code>-noreplay</code> parameter is not limited to use with <code>-force</code> but <code>-force</code> does require <code>-noreplay</code> if you do not want to replay in-flight transactions after you force the service to disconnect all sessions.
<code>-eval</code>	Use this parameter to hypothetically evaluate the impact of the command on the system.
<code>-verbose</code>	Verbose output.

Example

To temporarily relocate a named service member for the `crm` service from the database instance `crm1` to the database instance `crm3`:

```
$ srvctl relocate service -db crm -service crm -oldinst crm1 -newinst crm3
```

srvctl remove service

Removes the configuration for a service.

Syntax

```
srvctl remove service -db db_unique_name -service service_name
[-instance instance_name] [-global_override]
```

Parameters**Table A-83 srvctl remove service Command Parameters**

Parameter	Description
<code>-db db_unique_name</code>	Specify the unique name of the database on which the service that you want to remove operates.

Table A-83 (Cont.) srvctl remove service Command Parameters

Parameter	Description
<code>-service <i>service_name</i></code>	Specify the name of the service you want to remove.
<code>-instance <i>instance_name</i></code>	Optionally, you can specify the instance name of an administrator-managed database. Note: You can only use this parameter with Oracle Clusterware.
<code>-global_override</code>	Optionally, you can use this parameter to override value to operate on a global service. SRVCTL ignores this parameter a non-global service.

Examples

This following example removes the `sales` service from all instances of the clustered database named `crm`:

```
$ srvctl remove service -db crm -service sales
```

The following example removes the `sales` service from a specific instance of the `crm` clustered database:

```
$ srvctl remove service -db crm -service sales -instance crm02
```

srvctl start service

Starts a service or multiple services on a database, pluggable database, or instance.

Syntax

```
srvctl start service [-db db_unique_name] [-service "services_list"
  [-pq] | -pdb pluggable_database | -serverpool pool_name]
  [-node node_name | -instance instance_name]
  [-global_override] [-startoption start_options] [-eval] [-verbose]
```

Parameters

Parameter	Description
<code>-db <i>db_unique_name</i></code>	Specify a unique name for the database.
<code>-service "<i>service_list</i>"</code>	Specify a service name or a comma-delimited list of service names enclosed in double quotation marks (""). If you do not include this parameter, then SRVCTL starts all of the services for the specified database. Note: All manual service startup must specify the name of the service to be started by the user.
<code>-pq</code>	Specify this parameter to restrict the start action to a parallel query service.
<code>-pdb <i>pluggable_database</i></code>	Specify the name of a pluggable database. Optionally, you can specify either the name of a node or the name of an instance to restrict the starting of services to that particular object on the pluggable database.

Parameter	Description
<code>-serverpool pool_name</code>	Alternative to using the <code>-pq</code> parameter, you can specify the name of a server pool that contains the services you want to start. Use this parameter for policy-managed databases.
<code>-node node_name</code>	Specify the name of a node where the services reside that you want to start. Use this parameter for policy-managed databases.
<code>-instance instance_name</code>	Specify the name of an instance where the services reside that you want to start. Use this parameter for administrator-managed databases.
<code>-global_override</code>	Override value to operate on a global service. Use this parameter only with global services; this parameter is ignored if specified for a non-global service.
<code>-startoption start_options</code>	Specify startup options used when service startup requires starting a database instance. Options include <code>OPEN</code> , <code>MOUNT</code> , and <code>NOMOUNT</code> . Note: For multi-word startup options, such as <code>read only</code> and <code>read write</code> , separate the words with a space and enclose in double quotation marks (" "). For example, "read only".
<code>-verbose</code>	Display verbose output.

Usage Notes

- The `srvctl start service` command will fail if you attempt to start a service that is already running.
- 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 can move a service or change the status of a service on an instance with the `srvctl modify service` and `srvctl relocate service` commands.

Examples

The following example starts all services on a specific database:

```
$ srvctl start database -db myDB
```

The following examples start a list of services (optionally restricted to a parallel query services in the latter example) regardless of the pluggable database on which they may reside:

```
$ srvctl start database -db myDB -service "myServ01,myServ02"
$ srvctl start database -db myDB -service "myServ01,myServ02" -pq
```

The following example starts all services in a given server pool:

```
$ srvctl start database -db myDB -serverpool myServerPool
```

The following examples start all services on a given pluggable database, optionally restricted to a single node or a single instance in the latter two examples, respectively:

```
$ srvctl start service -db myDB -pdb myPDB1
$ srvctl start service -db myDB -pdb myPDB1 -node myRACNode01
$ srvctl start service -db myDB -pdb myPDB1 -instance myDB01
```

The following example starts all services, for a given database, on a given instance (for all pluggable databases):

```
$ srvctl start service -db myDB -instance myDB01
```

The following example start all services for a given database on a given node (for all pluggable databases):

```
$ srvctl start service -db myDB -node myRACNode01
```

The following examples start a list of services on a given node or given instance:

```
$ srvctl start service -db myDB -service "myService01,myService02" -node myRACNode01
$ srvctl start service -db myDB -service "myService01,myService02" -instance myDB01
```

srvctl status service

Displays the status of a service.

For Oracle RAC One Node databases, if there is an online database relocation in process, then this command displays the source and destination nodes and the status of the relocation, whether it is active or failed.

Syntax

```
srvctl status service -db db_unique_name [-service "service_name_list"]
[-force] [-verbose]
```

Parameters

Optionally, you can use this parameter to include disabled applications.

Table A-84 srvctl status service Command Parameters

Parameter	Description
-db <i>db_unique_name</i>	Specify the unique name of the database on which the service operates for which you want to check the status.
-service " <i>service_name_list</i> "	Optionally, you can specify a comma-delimited list of service names for which you want to check status. If you do not use this parameter, then SRVCTL lists the status of all services for the specified database.
-force	Optionally, you can use this parameter to include disabled applications.
-verbose	Optionally, you can use this parameter to display detailed output.

srvctl stop service

Stops one or more services globally across the cluster database, or on the specified instance.

Syntax

```
srvctl stop service {-node node_name | -db db_unique_name [-pq] [-pdb
pluggable_database |
  -service "service_list" [-eval]] [-node node_name | -instance instance_name |}
  -serverpool pool_name] [-stopoption IMMEDIATE|TRANSACTIONAL|NONE] [-drain_timeout
```

```
timeout]
  [-wait {YES | NO}] [-force [-noreplay]] [-global_override] [-verbose]
```

Parameters

Table A-85 srvctl stop service Command Parameters

Parameter	Description
-node <i>node_name</i>	Optionally, you can specify the name of the node on which you want to stop services. Use this parameter to stop all services on a specific node. Use this parameter without the <code>-db</code> parameter for both policy-managed and administrator-managed databases. If you use the <code>-db</code> parameter, then this only applies to policy-managed databases.
-db <i>db_unique_name</i>	Specify a unique name for the database.
-pdb <i>pluggable_database</i>	Alternatively, use this parameter to stop services running on a specific pluggable database.
-service " <i>service_list</i> "	Specify a particular service or a comma-delimited list of service names enclosed in double quotation marks (") you want to stop. If you do not provide a service name list, then SRVCTL stops all services on the database or on a specific instance.
-pq	Specify this parameter to restrict the stop action to a parallel query service.
-instance <i>instance_name</i>	Optionally, you can specify the name of the instance for which you want to stop services. Use this parameter for administrator-managed databases.
-serverpool <i>pool_name</i>	Optionally, you can specify the name of the server pool that contains the service you want to stop.
-eval	Use this parameter to hypothetically evaluate the impact of the command on the system.
-stopoption IMMEDIATE TRANSACTIONAL NONE	Optionally, you can specify the mode in which the service is stopped. If you set this parameter on the service, then this value is used if you do not set the value on the command line. IMMEDIATE terminates the sessions after the drain timeout interval expires. TRANSACTIONAL permits sessions to drain for the amount of time specified in the <code>-drain_timeout</code> parameter. The sessions are then terminated immediately when the drain timeout interval expires. If you specify NONE, then no sessions are terminated. The default is taken from the service setting, when specified on the service. Otherwise the default is NONE.
-drain_timeout <i>timeout</i>	Specify the time, in seconds, allowed for resource draining to be completed. Accepted values are an empty string (""), 0, or any positive integer. The default value is an empty string, which means that this parameter is not set. If it is set to 0, then draining occurs, immediately. The draining period is intended for planned maintenance operations. During the draining period, all current client requests are processed, but new requests are not accepted. When set on the service this value is used when the command line value is not set.

Table A-85 (Cont.) srvctl stop service Command Parameters

Parameter	Description
-wait YES NO	Choose YES to wait until service draining is completed on the node to stop the service.
-force [-noreplay]	Force SRVCTL to stop the service; this causes SRVCTL to disconnect all of the sessions using the stop option you specify (IMMEDIATE or TRANSACTIONAL), requiring the sessions using the service to reconnect and then connect to another instance. Notes: <ul style="list-style-type: none"> If you do not specify the <code>-force</code> parameter, then sessions already connected to this service stay connected, but new sessions cannot be established to the service. Optionally, you can specify the <code>-noreplay</code> parameter if you do not want Application Continuity to replay in-flight transactions after a session is terminated. The <code>-noreplay</code> parameter is not limited to use with <code>-force</code> but <code>-force</code> does require <code>-noreplay</code> if you do not want to replay in-flight transactions after you force the service to stop.
-global_override	Override value to operate on a global service. SRVCTL ignores this parameter if the service is not a global service.
-verbose	Use this parameter to display verbose output.

Example

The following example command stops services running on database db1:

```
$ srvctl stop service -db db1 -drain_timeout 60 -force - stopoption immediate -
verbose
```

srvctl add srvpool

Adds a server pool that is configured to host Oracle databases to a cluster.

Syntax

```
srvctl add srvpool -serverpool server_pool_name [-eval]
[-importance importance] [-min min_size] [-max max_size]
[-servers "node_list" | -category server_category] [-force] [-verbose]
```

Parameters**Table A-86 srvctl add srvpool Command Parameters**

Parameter	Description
-serverpool <i>server_pool_name</i>	The name of the server pool.
-eval	Use this parameter to hypothetically evaluate the impact of the command on the system.
-importance <i>importance</i>	The importance of the server pool (default value is 0).
-min <i>min_size</i>	The minimum size of the server pool (default value is 0).

Table A-86 (Cont.) srvctl add srvpool Command Parameters

Parameter	Description
<code>-max <i>max_size</i></code>	The maximum size of the server pool. The default value is -1, which indicates that the size is unlimited.
<code>-servers "<i>node_list</i>"</code>	A comma-delimited list of candidate node names enclosed in double quotation marks (" "). The server pool will only include nodes on the candidate list, but not all nodes on the candidate list will necessarily be in the server pool. Note: In Oracle Database 12c, servers are assigned to server pools according to the value of the <code>-category</code> parameter.
<code>-category <i>server_category</i></code>	The category of servers to use for the server pool, or "" for the empty category value. You can use <code>hub</code> for <code>ora.hub.category</code> , <code>leaf</code> for <code>ora.leaf.category</code> , or you can define your own category. The default is <code>ora.hub.category</code> .
<code>-force</code>	Add the server pool, even if it requires stopping resources in other server pools.
<code>-verbose</code>	Display verbose output.

Usage Notes

- SRVCTL prepends "ora." to the name of the server pool.
- This command is only available with Oracle Clusterware.

Example

The following command adds a server pool named SP1, with importance set to 1, the minimum number of nodes in the server pool set to 3 and the maximum number of nodes in the server pool set to 7:

```
srvctl add srvpool -serverpool SP1 -importance 1 -min 3 -max 7
```

srvctl config srvpool

Displays configuration information including name, minimum size, maximum size, importance, and a list of server names, if applicable, for a specific server pool in a cluster.

Syntax

```
srvctl config srvpool [-serverpool pool_name]
```

Parameters

The only parameter available for this command is `-serverpool pool_name`, which is the name of the server pool for which you want to display the configuration information.

Usage Notes

This command is only available with Oracle Clusterware.

Example

An example of this command is:

```
$ srvctl config srvpool -serverpool dbpool
```

srvctl modify srvpool

Modifies a server pool in a cluster.

If minimum size, maximum size, and importance are numerically increased, then the CRS daemon may attempt to reassign servers to this server pool, if by resizing, other server pools have comparatively lower minimum size and importance, to satisfy new sizes of this server pool.

Syntax

```
srvctl modify srvpool -serverpool pool_name [-eval] [-importance importance]
[-min min_size] [-max max_size] [-servers "server_list"]
[-category "server_category"] [-verbose] [-force]
```

Parameters

Table A-87 srvctl modify srvpool Command Parameters

Parameter	Description
-serverpool <i>pool_name</i>	Specify the name of the server pool you want to modify.
-eval	Optionally, you can use this parameter to hypothetically evaluate the impact of the command on the system.
-importance <i>importance</i>	Optionally, you can modify the importance of the server pool.
-min <i>min_size</i>	Optionally, you can modify the minimum size of the server pool. The default value is 0.
-max <i>max_size</i>	Optionally, you can modify the maximum size of the server pool. A value of -1 sets the server pool maximum size to UNLIMITED.
-servers " <i>server_list</i> "	Optionally, you can specify a comma-delimited list of candidate server names enclosed in double quotation marks (""). Note: In Oracle Database 12c, servers are assigned to server pools according to the value of the -category parameter.
-category " <i>server_category</i> "	Optionally, you can modify the server category enclosed in double quotation marks ("") or "" for empty category value.
-verbose	Optionally, you can use this parameter to display detailed output.
-force	Optionally, you can use this parameter to force the operation even though SRVCTL may stop some resources.

Usage Notes

You can only use this command with Oracle Clusterware.

Example

The following example changes the importance rank to 0, the minimum size to 2, and the maximum size to 4 for the server pool `srvpool1`:

```
$ srvctl modify srvpool -serverpool srvpool1 -importance 0 -min 2 -max 4
```


srvctl remove srvpool

Removes a specific server pool.

If there are databases or services that depend upon this server pool, then those resources are removed from the server pool first so that the remove server pool operation succeeds.

Syntax

```
srvctl remove srvpool -serverpool pool_name [-eval] [-verbose]
```

Parameters

Table A-88 srvctl remove srvpool Command Parameters

Parameter	Description
<code>-serverpool <i>pool_name</i></code>	Specify the name of the server pool you want to remove.
<code>-eval</code>	Optionally, you can use this parameter to evaluate the effects of removing a server pool without making any changes to the system.
<code>-verbose</code>	Optionally, you can use this parameter to display detailed output.

Usage Notes

- You can only use this command with Oracle Clusterware.
- If you successfully remove the specified server pool, then the CRS daemon may assign its servers to other server pools depending upon their minimum size, maximum size, and importance. The CRS daemon may also return these servers to its Free server pool.

Example

The following example removes a server pool from the system:

```
$ srvctl remove srvpool -serverpool srvpool1
```

srvctl status srvpool

Displays server pool names, number of servers in server pools, and, optionally, the names of the servers in the server pools.

Syntax

```
srvctl status srvpool [-serverpool pool_name] [-detail]
```

Usage Notes

- You can only use this command with Oracle Clusterware.
- Optionally, you can specify the name of a server pool for which you want to check the status. If you use this parameter, then the output includes the server pool name and number of servers in the server pool (and, optionally, the server names) for the specified server pool.

- If you choose to use the `-detail` parameter but do not specify a specific server pool with the `-serverpool` parameter, then the output of this command includes the names of servers that are currently assigned to each server pool.

srvctl add vip

Adds a virtual IP address (VIP) to a node.

Syntax

```
srvctl add vip -node node_name -address {VIP_name|ip}/netmask[/if1[|if2|...]]
-netnum network_number [-skip] [-verbose]
```

Parameters

Table A-89 srvctl add vip Command Parameters

Parameter	Description
<code>-node <i>node_name</i></code>	The name of the node on which you are adding the VIP.
<code>-address {<i>VIP_name</i> <i>ip</i>}/<i>netmask</i> [/<i>if1</i>[<i>if2</i> ...]]</code>	This specification creates a traditional VIP node application on the specified node. You can specify one <i>VIP_name</i> or address, along with an IPv4 netmask or IPv6 prefix length.
<code>-netnum <i>network_number</i></code>	The network number from which VIPs are obtained. The default network number is 1.
<code>-skip</code>	Specify this parameter to skip checking the reachability of the VIP address.
<code>-verbose</code>	Verbose output



Note:

Usage Notes

- You cannot have multiple VIPs on the same net number (subnet or interface pair) on the same node.
- This command is only available with Oracle Clusterware.

Example

An example of this command is:

```
# srvctl add network -netnum 2 -subnet 192.168.16.0/255.255.255.0
# srvctl add vip -node node7 -address 192.168.16.17/255.255.255.0 -netnum 2
```

The first command creates a network number, 2, and the second command adds a VIP to this network. You can specify the network number after the `-netnum` parameter in other SRVCTL commands.

srvctl config vip

Displays all VIPs on all networks in the cluster except for user VIPs.

Syntax

```
srvctl config vip {-node node_name | -vip vip_name}
```

Parameters

Table A-90 srvctl config vip Command Parameters

Parameter	Description
-node <i>node_name</i>	Specify the node name.
-vip <i>vip_name</i>	Alternatively, you can specify the VIP name.

Usage Notes

This command is only available with Oracle Clusterware.

Example

This command returns output similar to the following:

```
$ srvctl config vip -node crmnode1
```

```
VIP exists: ipv4, ipv6, network number 1, hosting node adc2100252
```

srvctl disable vip

Disables a specific VIP.

Syntax

```
srvctl disable vip -vip vip_name [-verbose]
```

Usage Notes

- This command is only available with Oracle Clusterware.
- Specify the name of the VIP you want to disable.
- Optionally, you can use the `-verbose` parameter to display detailed output.

Example

The following command disables a VIP:

```
$ srvctl disable vip -vip vip1 -verbose
```

srvctl enable vip

Enables a specific VIP.

Syntax

```
srvctl enable vip -vip vip_name [-verbose]
```

Usage Notes

- You can only use this command with Oracle Clusterware.
- Specify the name of the VIP you want to enable.
- Optionally, you can use the `-verbose` parameter to display detailed output.

Example

The following example enables a VIP named `crml-vip`:

```
$ srvctl enable vip -vip crml-vip -verbose
```

srvctl getenv vip

Obtains the values of environment variables for a specific VIP.

Syntax

```
srvctl getenv vip -vip vip_name [-envs "name_list"] [-verbose]
```

Parameters**Table A-91** `srvctl getenv vip` Command Parameters

Parameter	Description
<code>-vip vip_name</code>	Specify the name of the VIP for which you want to obtain the values of the environment variables.
<code>-envs "name_list"</code>	Optionally, you can specify a comma-delimited list of the names of specific environment variables. If you do not use this parameter, then SRVCTL displays the values of all environment variables associated with the VIP.

Usage Notes

You can only use this command with Oracle Clusterware.

Example

The following example lists all environment variables for the specified VIP:

```
$ srvctl getenv vip -vip nodel-vip
```

srvctl modify vip

Modifies IP address type but you can also use it to modify just the IP address.

Syntax

```
srvctl modify vip -node node_name -address {VIP_name|ip}/netmask[/if1[|if2|...]]
[-netnum network_number] [-verbose]
```

Parameters

Table A-92 `srvctl modify vip` Command Parameters

Parameter	Description
<code>-node node_name</code>	Specify the name of the node on which you are changing the VIP.
<code>-address {VIP_name ip}/netmask[/if1[if2 ...]]</code>	Use this parameter to change the configuration of an existing VIP. If the VIP has an IPv4 address and the address you specify is IPv6, and the IP address type is set to <code>both</code> and the network type is set to <code>static</code> , then SRVCTL adds the IPv6 address to the existing IPv4 address of that resource. You can specify one <code>VIP_name</code> or IP address, along with an IPv4 netmask or IPv6 prefix length.
<code>-netnum network_number</code>	Optionally, you can specify the network number from which VIPs are obtained. If you do not use this parameter, then SRVCTL obtains the VIPs from the same default network from which the <code>nodeapps</code> VIP is obtained.
<code>-verbose</code>	Optionally, you can use this parameter to display detailed output.

Usage Notes

- You can only use this command with Oracle Clusterware.
- You cannot have multiple VIPs on the same net number (subnet or interface pair) on the same node.

Example

The following example adds an IPv4 address to a VIP, if one does not already exist. If the VIP has an IPv4 address, then it is replaced with the new network specification.

```
# srvctl modify vip -node node7 -address 192.168.16.17/255.255.255.0 -netnum 2
```

srvctl predict vip

Predicts the consequences of VIP failure.

Syntax

```
srvctl predict vip [-vip vip_name] [-verbose]
```

Usage Notes

- Optionally, you can specify the name of a VIP for which you want to evaluate the consequences of failure.
- Optionally, you can use the `-verbose` parameter to display detailed output.

srvctl relocate vip

Relocates a specific VIP from its current hosting node to another node within the cluster.

Syntax

```
srvctl relocate vip -vip vip_name [-node node_name] [-force] [-verbose]
```

Parameters

Table A-93 srvctl relocate vip Command Parameters

Parameter	Description
-vip vip_name	Specify the name of the VIP you want to relocate.
-node node_name	Optionally, you can specify the name of the target node where you want to relocate the VIP.
-force	Optionally, you can use this parameter to force the relocation of the VIP regardless of any dependencies.
-verbose	Optionally, you can use this parameter to display detailed output.

Example

The following example relocates a VIP to a different node in the cluster:

```
$ srvctl relocate vip -vip vip1 -node node3
```

srvctl remove vip

Removes specific VIPs.

Syntax

```
srvctl remove vip -vip "vip_name_list" [-force] [-noprompt] [-verbose]
```

Parameters

Table A-94 srvctl remove vip Command Parameters

Parameter	Description
-vip "vip_name_list"	Specify a comma-delimited list of VIP names that you want to remove surrounded by double quotation marks (" ").
-force	Optionally, you can use this parameter to remove a VIP regardless of any dependencies.
-noprompt	Optionally, you can use this parameter to suppress prompts.
-verbose	Optionally, you can use this parameter to display detailed output.

Usage Notes

You can only use this command with Oracle Clusterware.

Example

The following example removes several VIPs from the system:

```
$ srvctl remove vip -vip "vip1,vip2,vip3" -force -noprompt -verbose
```

srvctl setenv vip

Administers cluster VIP environment configurations.

Syntax

```
srvctl setenv vip -vip vip_name {-envs "name=val[,...]" | -env "name=val"}  
[-verbose]
```

Parameters

Table A-95 srvctl setenv vip Command Parameters

Parameter	Description
-vip vip_name	Specify the name of the VIP for which you want to set environment variables.
-envs "name=val[,...]"	Specify a comma-delimited list of name-value pairs of environment variables enclosed in double quotation marks (") that you want to set.
-env "name=val"	Alternative to a list of environment variables, you can use this parameter to set a single environment variable to a value that contains commas or other special characters enclosed in double quotation marks (").
-verbose	Optionally, you can use this parameter to display detailed output.

Usage Notes

You can only use this command with Oracle Clusterware.

Example

The following example sets the language environment configuration for a cluster VIP:

```
$ srvctl setenv vip -vip crml-vip -env "LANG=en"
```

srvctl start vip

Starts a specific VIP or a VIP on a specific node.

Syntax

```
srvctl start vip {-node node_name | -vip vip_name} [-verbose]
```

Parameters

Table A-96 srvctl start vip Command Parameters

Parameter	Description
-node node_name	Specify the name of the node on which the VIP resides that you want to start.
-vip vip_name	Alternative to specifying a node, you can specify a VIP that you want to start.

Table A-96 (Cont.) srvctl start vip Command Parameters

Parameter	Description
-verbose	Optionally, you can use this parameter to display detailed output.

Usage Notes

You can only use this command with Oracle Clusterware.

Example

The following example starts a specific VIP:

```
$ srvctl start vip -vip crml-vip -verbose
```

srvctl status vip

Displays status for a specific VIP or a VIP on a specific node.

Syntax

```
srvctl status vip {-node node_name | -vip vip_name} [-verbose]
```

Parameters**Table A-97 srvctl status vip Command Parameters**

Parameter	Description
-node <i>node_name</i>	Specify the name of the node on which the VIP resides that you want to check the status.
-vip <i>vip_name</i>	Alternative to specifying a node, you can specify a VIP that you want to check the status.
-verbose	Optionally, you can use this parameter to display detailed output.

Usage Notes

You can only use this command with Oracle Clusterware.

srvctl stop vip

Stops a specific VIP or all VIPs on a specific node, including any VIPs that were relocated due to a failover.

Syntax

```
srvctl stop vip {-node node_name | -vip vip_name} [-force] [-relocate] [-verbose]
```


Parameters

Table A-98 `srvctl stop vip` Command Parameters

Parameter	Description
<code>-node node_name</code>	Specify the name of a node on which a VIP resides that you want to stop. If you use this parameter, then SRVCTL stops all VIPs on the specific node, including failed-over VIPs.
<code>-vip vip_name</code>	Alternative to specifying a node, you can specify a VIP that you want to stop.
<code>-force</code>	Optionally, you can use this parameter to stop the VIP regardless of any dependencies.
<code>-relocate</code>	Optionally, you can use this parameter to relocate the VIP. Note: You must use the <code>-node node_name</code> parameter with the <code>-relocate</code> parameter.
<code>-verbose</code>	Optionally, you can use this parameter to display detailed output.

Usage Notes

You can only use this command with Oracle Clusterware.

Example

The following example stops all the VIPs on `mynode1`, including any failed-over VIPs:

```
$ srvctl stop vip -node mynode1 -verbose
```

srvctl unsetenv vip

Unsets the environment configuration for the specified cluster VIP.

Syntax

```
srvctl unsetenv vip -vip "vip_name_list" -envs "name_list" [-verbose]
```

Parameters

Table A-99 `srvctl unsetenv vip` Command Parameters

Parameter	Description
<code>-vip "vip_name_list"</code>	Specify a comma-delimited list of VIP names enclosed in double quotation marks (" ").
<code>-envs "name_list"</code>	Specify a comma-delimited list of environment variable names enclosed in double quotation marks (" ") that you want to unset.
<code>-verbose</code>	Optionally, you can use this parameter to display detailed output.

Example

The following example unsets the `CLASSPATH` environment variable for a cluster VIP:

```
$ srvctl unsetenv vip -vip "crm2-vip" -envs "CLASSPATH"
```

srvctl config volume

Displays the configuration for a specific volume or all volumes.

Syntax

```
srvctl config volume [-volume volume_name] [-diskgroup disk_group_name]  
                    [-device volume_device]
```

Parameters

Table A-100 srvctl config volume Command Parameters

Parameter	Description
-volume <i>volume_name</i>	Specify the name of the volume for which you want to view the configuration.
-diskgroup <i>disk_group_name</i>	Specify the name of the disk group in which the volume resides for which you want to display the configuration.
-device <i>volume_device</i>	Specify the path to the volume device for which you want to display the configuration.

Usage Notes

- If you do not specify any of the optional parameters, then SRVCTL displays the configuration information for all volumes.
- If you specify only the `-volume` parameter, then SRVCTL displays the configuration for all volumes with that name, regardless of the diskgroup.
- If you specify only the `-diskgroup` parameter, then SRVCTL displays the configuration information for the volumes that reside in the disk group that you specify.
- If you specify only the `-device` parameter, then SRVCTL displays the configuration information for the volume matching that device specifier.
- If you specify the `-diskgroup` and `-device` parameters, then SRVCTL displays the configuration information for the volume device that resides in the disk group that you specify.
- This command is only available with Oracle Clusterware.

Examples

This command returns information similar to the following:

```
$ srvctl config volume -device /dev/asm/volume1-123
```

```
Diskgroup Name: DGL  
Volume Name   : VOL1  
Volume Device : /dev/asm/volume1-123  
Volume is enabled.  
Volume is enabled on nodes:  
Volume is disabled on nodes:
```

If you do not specify any parameters, then SRVCTL returns configuration information for all volumes, similar to the following:

```
$ srvctl config volume

Diskgroup name: DG1
Volume name: VOL1
Volume device: /dev/asm/volume1-123
Volume is enabled.
Volume is enabled on nodes:
Volume is disabled on nodes:
Diskgroup name: DG1
Volume name: VOL2
Volume device: /dev/asm/volume2-456
Volume is enabled.
Volume is enabled on nodes:
Volume is disabled on nodes:
```

srvctl disable volume

Disables Oracle Clusterware management for a specific volume or all volumes. This command allows a volume device to be stopped by operating on the Oracle Clusterware resource for the volume. This command does not stop volume device.

Syntax

```
srvctl disable volume {-volume volume_name -diskgroup disk_group_name |
    -device volume_device}
```

Parameters

Table A-101 srvctl disable volume Command Parameters

Parameter	Description
-volume <i>volume_name</i>	Specify the name of the volume that you want to disable.
-diskgroup <i>disk_group_name</i>	Specify the name of the disk group in which the volume that you want to disable resides.
-device <i>volume_device</i>	Alternative to using the -diskgroup parameter, you can specify the path to the volume device that you want to disable.

Usage Notes

- You can only use this command with Oracle Clusterware.
- You must specify a particular volume that you want to disable. You can specify a volume that resides in either a particular disk group or on a particular volume device.

Example

The following example disables a volume named `VOLUME1` that resides in a disk group named `DATA`:

```
$ srvctl disable volume -volume VOLUME1 -diskgroup DATA
```

srvctl enable volume

Enables Oracle Clusterware management for a specific volume or all volumes. This command allows a volume device to be started by operating on the Oracle Clusterware resource for the volume. This command does not start the volume device, and is different from the SQL command `ALTER DISKGROUP ENABLE VOLUME` or the ASMCMD command `volenable`, because these two commands bring the volume device online, in a running state, making the volume device accessible.

Syntax

```
srvctl enable volume {-volume volume_name -diskgroup disk_group_name |
                    -device volume_device}
```

Parameters

Table A-102 srvctl enable volume Command Parameters

Parameter	Description
<code>-volume volume_name</code>	Specify the name of the volume that you want to enable.
<code>-diskgroup disk_group_name</code>	Specify the name of the disk group in which the volume that you want to enable resides.
<code>-device volume_device</code>	Alternative to using the <code>-diskgroup</code> parameter, you can specify the path to the volume device that you want to enable.

Usage Notes

- You can only use this command with Oracle Clusterware.
- You must specify a particular volume that you want to enable. You can specify a volume that resides in either a particular disk group or on a particular volume device.

Example

The following example enables a volume named `VOLUME1` that resides in a disk group named `DATA`:

```
$ srvctl enable volume -volume VOLUME1 -diskgroup DATA
```

srvctl remove volume

Removes a specific volume.

Syntax

Use this command with one of the following syntax models:

```
srvctl remove volume -volume volume_name -diskgroup disk_group_name [-force]
```

```
srvctl remove volume -device volume_device [-force]
```

Parameters

Table A-103 `srvctl remove volume` Command Parameters

Parameter	Description
<code>-volume volume_name</code>	Specify the name of the volume that you want to remove.
<code>-diskgroup disk_group_name</code>	Specify the name of the disk group in which the volume that you want to remove resides.
<code>-device volume_device</code>	Specify the path to the file system resource in which the volume that you want to remove resides.
<code>-force</code>	You can use this parameter to remove the volume even if it is running.

Usage Notes

- You can only use this command with Oracle Clusterware.
- The volume gets created when you create volumes in Oracle ASM.
- You must specify a particular volume that you want to remove. You can specify a volume that resides in either a particular disk group or on a particular volume device.

Example

The following example removes a volume named `VOLUME1` that resides in a disk group named `DATA`:

```
$ srvctl remove volume -volume VOLUME1 -diskgroup DATA
```

srvctl start volume

Starts a specific, enabled volume.

Syntax

```
srvctl start volume {-volume volume_name -diskgroup disk_group_name |  
-device volume_device} [-node node_list]
```

Parameters

Table A-104 `srvctl start volume` Command Parameters

Parameter	Description
<code>-volume volume_name</code>	Specify the name of the volume that you want to start.
<code>-diskgroup disk_group_name</code>	Specify the name of the disk group in which the volume that you want to start resides.
<code>-device volume_device</code>	Alternative to using the <code>-diskgroup</code> parameter, you can specify the path to the volume device that you want to start.
<code>-node node_list</code>	Optionally, you can specify a comma-delimited list of node names enclosed in double quotation marks (" ") where volumes that you want to start reside.

Usage Notes

- You can only use this command with Oracle Clusterware.
- The `srvctl start volume` command does not create a volume service. Provided that the volume already exists and the volume resource is enabled, SRVCTL attempts to start it. If the volume exists but the resource is disabled, then `srvctl start volume` returns an error.

Example

The following example starts a volume named `VOLUME1` that resides in a disk group named `DATA`:

```
$ srvctl start volume -volume VOLUME1 -diskgroup DATA
```

srvctl status volume

Displays the status of a specific volume or all volumes.

Syntax

```
srvctl status volume [-device volume_device] [-volume volume_name]
  [-diskgroup disk_group_name] [-node "node_list"]
```

Parameters

Table A-105 `srvctl status volume` Command Parameters

Parameter	Description
<code>-device volume_device</code>	Optionally, you can specify the path to the volume device for which you want to display the status.
<code>-volume volume_name</code>	Optionally, you can specify the name of the volume for which you want to view the status.
<code>-diskgroup disk_group_name</code>	Optionally, you can specify the name of the disk group in which the volume resides for which you want to display the status.
<code>-node "node_list"</code>	Optionally, you can specify a comma-delimited list of node names enclosed in double quotation marks (" ") where volumes for which you want to view the status reside.

Usage Notes

- You can only use this command with Oracle Clusterware.
- If you do not specify any of the optional parameters, then SRVCTL displays the status for all volumes.
- If you specify only the `-volume` parameter, then SRVCTL displays the status for the volume that you specify.
- If you specify only the `-diskgroup` parameter, then SRVCTL displays the status for the volumes that reside in the disk group that you specify.
- If you specify only the `-device` parameter, then SRVCTL displays the status for the volume device that you specify.

- If you specify the `-diskgroup` and `-device` parameters, then SRVCTL displays the status for the volume device in the disk group that you specify.
- If you specify the `-node` parameter, then SRVCTL displays the status of the volumes that reside on the nodes you list.

Examples

This command displays information similar to the following:

```
$ srvctl status volume -volume vol1
Volume vol1 of diskgroup diskgrp1 for device volume_device_path1 is enabled
Volume vol1 of diskgroup diskgrp1 for device volume_device_path1 is running
```

In the preceding example, SRVCTL performs a status query on all nodes because the `-node` parameter is not specified.

```
$ srvctl status volume
Volume vol1 of diskgroup diskgrp for device volume_device_path1 is enabled
Volume vol1 of diskgroup diskgrp for device volume_device_path1 is running
Volume vol2 of diskgroup diskgrp for device volume_device_path2 is enabled
Volume vol2 of diskgroup diskgrp for device volume_device_path2 is running
```

In the preceding example, SRVCTL displays the status of all registered volumes because the `no` parameter is specified.

srvctl stop volume

Stops a specific, running volume.

Syntax

```
srvctl stop volume {-volume volume_name -diskgroup disk_group_name |
  -device volume_device} [-node "node_list"]
```

Parameters

Table A-106 srvctl stop volume Command Parameters

Parameter	Description
<code>-volume volume_name</code>	Specify the name of the volume you want to stop.
<code>-diskgroup disk_group_name</code>	Specify the name of the disk group in which the volume you want to stop resides.
<code>-device volume_device</code>	Alternative to using the <code>-diskgroup</code> parameter, you can specify the path to the volume device that you want to stop.
<code>-node "node_list"</code>	Optionally, you can specify a comma-delimited list of node names enclosed in double quotation marks (" ") where volumes that you want to stop reside.

Usage Notes

- You can only use this command with Oracle Clusterware.
- The `srvctl stop volume` command attempts to stop (disable) the volume but it does not disable the resource or remove the volume from Oracle ASM.

Example

The following example stops a volume named `VOLUME1` that resides in a disk group named `DATA`:

```
$ srvctl stop volume -volume VOLUME1 -diskgroup DATA
```


B

Troubleshooting Oracle RAC

This appendix explains how to 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 RAC](#)
- [Using Instance-Specific Alert Files in Oracle RAC](#)
- [Enabling Tracing for Java-Based Tools and Utilities in Oracle RAC](#)
- [Resolving Pending Shutdown Issues](#)
- [How to Determine If Oracle RAC Instances Are Using the Private Network](#)
- [Database Fails to Start after Private NIC Failure](#)

Note:

Trace and log files, similar to those generated for Oracle Database with Oracle RAC, are also available for the Oracle Clusterware components. For Oracle Clusterware, Oracle Database 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 Database 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 noncluster 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. For example:

```
$ORACLE_BASE/diag/rdbms/$DBNAME/$SID_NAME/trace
```

Oracle Database 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. 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 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 and *Oracle Database 2 Day + Real Application Clusters Guide* for more information about monitoring errors and alerts in trace files

Trace files are also 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 RAC

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 12c release 2 (12.2) 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 RAC

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` initialization parameter.

Enabling Tracing for Java-Based Tools and Utilities in Oracle RAC

All Java-based tools and utilities that are available in Oracle RAC are called 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), and the Server Control Utility (SRVCTL). For example, to run DBCA, enter the command `dbca`.

By default, Oracle Database enables traces for DBCA and the Database Upgrade Assistant (DBUA). For the CVU, and SRVCTL, you can set the `SRVM_TRACE` environment variable to `TRUE` to make Oracle Database generate traces. Oracle Database writes traces to log files. For example, Oracle Database writes traces to log files in `Oracle_base/cfgtoollogs/dbca` and `Oracle_base/cfgtoollogs/dbua` for DBCA and DBUA, respectively.

Resolving Pending Shutdown Issues

In some situations a `SHUTDOWN IMMEDIATE` may be pending and Oracle Database 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 the Oracle Enterprise Manager Cloud Control graphical user interface (GUI) to check the interconnect. Also, see the *Oracle Database 2 Day + Real Application Clusters Guide* for more information about monitoring Oracle RAC using Oracle 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

Database Fails to Start after Private NIC Failure

In a two-node cluster that is not using grid interprocess communication (GIPC) and has only a single private network interface card (NIC), when the NIC fails on the node on which the Oracle Grid Infrastructure Management Repository database (mgmtldb) is running (node A, for example) after the other node (node B) was evicted, then when the private network is reestablished on node A, node B joins the cluster but the database instance on node A is unable to start.

To start the database instance:

1. Stop the Oracle Clusterware stack on both nodes, as follows:

```
crsctl stop cluster
```

2. Start the Oracle Clusterware stack on both nodes, as follows:

```
crsctl start cluster
```

3. Stop the database instance on node A, on which the NIC failed, as follows:

```
srvctl stop instance -db db_unique_name -node A
```

4. Start the database on both nodes by running the following commands one of the nodes:

```
srvctl start instance -db db_unique_name -node B
srvctl start instance -db db_unique_name -node A
```

The database instance will fail to start on node A until after the instance starts on node B.

Glossary

Automatic Workload Repository (AWR)

A built-in repository that exists in every Oracle database. At regular intervals, Oracle Database makes a snapshot of all of its vital statistics and workload information and stores them in the AWR.

administrator-managed database

A database that you specifically define on which servers it can run, and where services can run within the database.

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 RAC that provides copies of blocks directly from a holding instance's memory cache to a requesting instance's memory cache.

cardinality

The number of database instances you want running during normal operations.

CDB

A multitenant container database (CDB) is an Oracle database that includes zero, one, or many user-created pluggable databases (PDBs). Every Oracle database is either a CDB or a non-CDB.

client cluster

A cluster that advertises its names with the [server cluster](#).

cluster

Multiple interconnected computers or servers that appear as if they are one server to end users and applications.

cluster configuration policy

A document that contains exactly one definition for each server pool defined in the system.

cluster configuration policy set

A document that defines the names of all server pools configured in the cluster and contains one or more configuration policies.

cluster database

The generic term for a Oracle RAC database.

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 Ready Services Daemon (CRSD)

The primary Oracle Clusterware process that performs high availability recovery and management operations, such as maintaining 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.

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 Time Synchronization Service

A time synchronization mechanism that ensures that all internal clocks of all nodes in a cluster are synchronized.

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.

commit outcome

A message sent to the client from the Oracle database after a transaction has been committed. These messages are not durable.

database pool

A set of databases within a database cloud that provide a unique set of global services and belong to a certain administrative domain. Partitioning of cloud databases into multiple pools simplifies service management and provides higher security because each pool can be administered by a different administrator.

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.

dynamic network

A network that uses DHCP for IPv4 or stateless autoconfiguration (autoconfig) for IPv6.

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.

failure group

A failure group is a subset of the disks in a disk group, which could fail at the same time because they share hardware. Failure groups are used to store mirror copies of data.

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 Database 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 Database gives the client a connection to an active instance that provides the requested database service.

file system

A file system is a software component providing structured access to disks. File systems present objects, such as files, to application programs. Access to files is generally specified with standard API defining operating system calls such as `Open/Close` and `Read/Write` that the application program uses for accessing files. File systems are usually provided as a component of an operating system, but may be provided as an independent software component.

forced disk write

In Oracle RAC, 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.

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.

Global Cache Service (GCS)

Process that implements 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.FOO.COM

global dynamic performance views (GV\$)

Dynamic performance views storing information about all open instances in an Oracle RAC 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. GSD is installed on the nodes by default.

Grid Plug and Play Daemon (GPNPD)

This process provides access to the Grid Plug and Play profile, and coordinates updates to the profile among the nodes of the cluster to ensure that all of the nodes have the most recent profile.

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).

high availability

Systems with redundant components that provide consistent and uninterrupted service, even if there are hardware or software failures. This involves some degree of redundancy.

Hub Node

A node in an [Oracle Flex Cluster](#) that is tightly connected with other servers and has direct access to shared storage.

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 Database allocates a memory area called the System Global Area (SGA) and starts one or more Oracle Database processes. This combination of the SGA and the Oracle Database processes is called an instance. Each instance has unique Oracle System Identifier (SID), instance name, rollback segments, and thread ID.

instance caging

A method that uses an initialization parameter to limit the number of CPUs that an instance can use simultaneously for foreground processes.

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 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.

keystore

A container that stores a Transparent Data Encryption key. In previous releases, this was referred to as a wallet.

Leaf Node

Servers that are loosely coupled with [Hub Nodes](#), which may not have direct access to the shared storage.

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.

memory pressure

A state indicating that there is a limited amount of available memory on a server.

metric

The rate of change in a cumulative statistic.

multitenant container database

See [CDB](#).

mutables

Non-deterministic functions that can change their results each time they are called. Mutable functions can cause replay to be rejected, if the function results change at replay. Consider `sequence.nextval` and `SYSDATE` used in key values. If a primary key is built with values from these function calls, and is used in later foreign keys or other binds, then, at replay the same function result must be returned. Application Continuity provides mutable value replacement at replay for granted Oracle function calls to provide opaque bind-variable consistency.

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 Oracle RAC and Oracle Clusterware software are installed.

Object Link Manager (OLM)

The Oracle interface that maps symbolic links to logical drives and displays them in the OLM graphical user interface.

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 Cluster File Systems

Oracle offers two cluster file systems, OCFS2 for Linux and Oracle ASM Cluster File System (Oracle ACFS). While Oracle ACFS is a proprietary file system, the source for OCFS2 for Linux is available to all under GNU's General Public License (GPL). The two file systems are not compatible.

Oracle Cluster Registry (OCR)

The Oracle RAC configuration information repository that manages information about the cluster node list and instance-to-node mapping information. OCR also manages information about Oracle Clusterware resource profiles for customized applications.

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 Extended Cluster

A cluster consist of nodes that are located in multiple locations called sites.

Oracle Flex Cluster

Large clusters that are made of up of two types of nodes: [Hub Nodes](#) and [Leaf Nodes](#), where the Hub Nodes form a cluster using current membership algorithms and Leaf Nodes connect for membership to a single Hub Node.

Oracle Grid Infrastructure

The software that provides the infrastructure for an enterprise grid architecture. In a cluster this software includes Oracle Clusterware and Oracle Automatic Storage Management (Oracle ASM). For a standalone server, this software includes Oracle Restart and Oracle ASM. Oracle Database combines these infrastructure products into one software installation called the *Oracle Grid Infrastructure home* (`Grid_home`).

Oracle Grid Naming Service Daemon (GNSD)

The Oracle Grid Naming Service is a gateway between the cluster mDNS and external DNS servers. The `gnsd` process performs name resolution within the cluster.

Oracle High Availability Services Daemon (OHASD)

This process anchors the lower part of the Oracle Clusterware stack, which consists of processes that facilitate cluster operations.

Oracle Interface Configuration Tool (OIFCFG)

A command-line tool for both noncluster 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 also uses OIFCFG to identify and display available interfaces.

Oracle Managed Files

A service that automates naming, location, creation, and deletion of database files such as control files, redo log files, data files and others, based on a few initialization parameters. You can use Oracle Managed Files on top of a traditional file system supported by the host operating system, for example, VxFS or ODM. It can simplify many aspects of the database administration by eliminating the need to devise your own policies for such details.

Oracle Notification Service

A publish and subscribe service for communicating information about all FAN events.

Oracle Universal Installer

A tool to install Oracle Clusterware, the Oracle relational database software, and the Oracle RAC software. You can also use the Oracle Universal Installer to launch the Database Configuration Assistant (DBCA).

Oracle XA

An external interface that allows global transactions to be coordinated by a transaction manager other than Oracle Database.

PDB

In a multitenant container database (CDB), a portable collection of schemas, schema objects, and nonschema objects that appears to an Oracle Net client as a non-CDB.

pluggable database

See [PDB](#).

policy-managed database

A database that you define as a cluster resource. Management of the database is defined by how you configure the resource, including on which servers the database can run and how many instances of the database are necessary to support the expected workload.

raw device

A disk drive that does not yet have a file system set up. Raw devices are used for Oracle RAC 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.

recoverable error

A class of errors that arise due to an external system failure, independent of the application session logic that is executing. Recoverable errors occur following planned and unplanned outages of foregrounds, networks, nodes, storage, and databases. The application receives an error code that can leave the application not knowing the status of the last operation submitted.

Recovery Manager (RMAN)

An Oracle tool that enables you to back up, copy, restore, and recover data files, control files, and archived redo logs. It is included with the Oracle server and does not require separate installation. You can run RMAN as a command line utility from the operating system (O/S) prompt or use the GUI-based Oracle Enterprise Manager Backup Manager.

region

A logical boundary that contains database clients and servers that are considered to be close to each other.

request

A unit of work submitted from the application. A request typically corresponds to the SQL and PL/SQL, and other database calls of a single web request, on a single database connection, and generally is demarcated by the calls made to check-out and check-in the database connection from a connection pool.

request boundary

A request boundary marks where an application or application server borrows and returns connections from their database connection pools.

result cache

A result cache is an area of memory, either in the SGA or client application memory, that stores the result of a database query or query block for reuse. The cached rows are shared across statements and sessions unless they become stale.

Runtime Connection Load Balancing

Enables Oracle Database 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 Oracle RAC 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 Utility (SRVCTL)

Server Management (SRVM) comprises the components required to operate Oracle Enterprise Manager in Oracle RAC. 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.

server

A computer system that has no Oracle software installed upon it.

server cluster

The cluster in which the shared GNS server is running.

server group

A logical partition of nodes in a cluster into a group that hosts applications, databases, or both. Server groups can be members of other server groups.

service level

A measure of the performance of a system.

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.

session state consistency

After a COMMIT has executed, if the state was changed in that transaction, then it is not possible to replay the transaction to reestablish that state if the session is lost. When configuring Application Continuity, the applications are categorized depending on whether the session state after the initial setup is dynamic or static, and then whether it is correct to continue past a COMMIT operation within a request.

- **Dynamic:** (default) A session has dynamic state if the session state changes are not fully encapsulated by the initialization, and cannot be fully captured in a callback at failover. Once the first transaction in a request commits, failover is internally disabled until the next request begins. This is the default mode that most applications should use for requests.
- **Static:** (special—on request) A session has a static state if all session state changes, such as NLS settings and PL/SQL package state, can be repeated in an initialization callback. This setting is used only for database diagnostic applications that do not change session state inside requests. Do not set STATIC mode if there are any non-transactional state changes in the request that cannot be reestablished by a callback. If you are unsure, use DYNAMIC mode.

shared everything

A database architecture in which all instances share access to all of the data.

single client access name (SCAN)

Oracle Database 11g database clients use SCAN to connect to the database. SCAN can resolve to multiple IP addresses, reflecting multiple listeners in the cluster handling public client connections.

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.

SQL translation profile

A SQL translation profile is a database schema object that directs how SQL statements in non-Oracle databases are translated to Oracle, and how Oracle error codes and ANSI SQLSTATES are translated into other vendors' equivalents.

system identifier (SID)

The Oracle system identifier (SID) identifies a specific instance of the running Oracle software. For an Oracle RAC 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 Oracle RAC and Oracle RAC 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 library.

Virtual Internet Protocol (VIP)

An IP address assigned to multiple applications residing on a single server, multiple domain names, or multiple servers, rather than being assigned to a specific single server or network interface card (NIC).

volume manager

A volume manager is a software component that manages the mapping of the collection of the pieces of the disks into a volume.

voting disk

A file that manages information about node membership.

Index

Symbols

\$ORACLE_HOME/root.sh script, [9-3](#)

A

abnormal instance termination, [3-9](#)

ACMS

Atomic Controlfile to Memory Service, [1-20](#)

Active Session History

Oracle RAC, [14-7](#)

Top Cluster Events, [14-7](#)

Top Remote Instance, [14-7](#)

active sessions, [14-7](#)

ACTIVE_INSTANCE_COUNT initialization

parameter, [3-18](#)

adding nodes to an existing cluster, [10-1](#)

adding Oracle RAC to nodes on Linux and UNIX,
[11-1](#)

adding Oracle RAC to nodes on Windows, [12-1](#)

ADDM

global monitoring, [14-6](#)

see Automatic Database Diagnostic Monitor,
[14-8](#)

ADDM for Oracle Real Application Clusters

mode, [14-6](#)

administering

services, [5-31](#)

services with SRVCTL, [5-33](#)

administering instances

with Server Management, [3-4](#)

administering Oracle Enterprise Manager jobs,
[3-31](#)

administering services

Oracle Enterprise Manager, [5-30](#)

SRVCTL, [5-30](#)

administrative tools

overview and concepts, [1-33](#)

administrator-managed database, [3-1](#)

relocating services in, [5-19](#)

administrator-managed database instances

adding, [12-3](#)

administrator-managed databases, [1-24](#), [5-27](#)

AVAILABLE instances for services, [5-27](#)

converting to policy-managed, [3-23](#)

administrator-managed databases (*continued*)

PREFERRED instances for services, [5-27](#)

ADRCI

ADR Command-Line Interpreter, [B-2](#)

Advanced Queuing

and FAN, [6-2](#)

Advisor Central in Oracle Enterprise Manager,
[14-6](#)

affinity, [5-19](#)

aggregates

by instances, [14-3](#)

by services, [14-3](#)

by waits, [14-3](#)

alert administration

Oracle Enterprise Manager, [3-31](#)

alert blackouts

Oracle Enterprise Manager, [3-32](#)

alert logs, [B-3](#)

managing, [B-1](#)

ALTER SYSTEM ARCHIVE LOG CURRENT

statement, [3-6](#)

ALTER SYSTEM ARCHIVE LOG statement, [3-6](#)

INSTANCE option, [3-6](#)

ALTER SYSTEM CHECKPOINT statement

global versus local, [3-6](#)

specifying an instance, [3-6](#)

ALTER SYSTEM KILL SESSION statement

terminating a session on a specific instance,
[3-13](#)

ALTER SYSTEM QUIESCE RESTRICTED

statement

quiescing a noncluster database, [3-25](#)

ALTER SYSTEM statement

CHECKPOINT clause, [3-6](#)

ALTER SYSTEM SWITCH LOGFILE statement,

[3-6](#)

Application Continuity, [6-1](#), [6-21](#)

auto session state consistency, [6-47](#)

commit outcome, [6-21](#)

concepts, [6-21](#)

configuring, [6-32](#)

configuring service attributes, [5-34](#)

database request, [6-21](#)

delaying reconnection, [6-37](#)

described, [6-26](#)

Application Continuity (*continued*)
 dynamic session state consistency, [6-47](#)
 establishing initial state, [6-34](#)
 introduction to, [1-21](#)
 mutable functions, [6-21](#)
 potential side effects, [6-50](#)
 protection-level statistics, [6-46](#)
 recoverable error, [6-21](#)
 restoring state settings, [6-35](#)
 restrictions, [6-51](#)
 running without, [6-40](#)
 session state consistency, [6-21](#)
 static session state consistency, [6-49](#)
 using for planned maintenance, [6-39](#)

applications
 consolidating multiple in a single database,
[13-3](#)
 highly available, [13-1](#)
 scalability, [13-5](#)
 spanning XA transactions across Oracle
 RAC instances, [5-16](#)
 using pre-created database sessions, [5-13](#)

ARCHIVE LOG command
 SQL*Plus, [3-6](#)

archive logs
 destinations, converting to cluster database,
[15-1](#)

ARCHIVE_LAG_TARGET initialization
 parameter, [3-22](#)

archived redo log files
 applying in parallel, [8-7](#)
 file format and destination, [7-5](#)
 log sequence number, [7-5](#)

archiver process
 monitor, [7-10](#)

archiving mode
 changing, [7-1](#)

ASH reports, [14-7](#)

ASM_PREFERRED_READ_FAILURE_GROUPS
 initialization parameter, [2-6](#), [3-18](#)

Atomic Controlfile to Memory Service (ACMS),
[1-20](#)

Automatic Database Diagnostic Monitor (ADDM),
[1-34](#), [14-6](#), [14-8](#), [14-9](#), [14-11](#)
 analyzing AWR data, [14-6](#)
 DBMS_ADDM PL/SQL package, [14-6](#)
 DBMS_ADVISOR PL/SQL package, [14-6](#)
 Global ADDM mode, [14-6](#)
 Local ADDM mode, [14-6](#)

Automatic Diagnostic Repository (ADR), [B-2](#)
 ADRCI command-line interpreter, [B-2](#)

automatic load balancing
 configuring RMAN channels for multiple
 instances, [7-3](#)

AUTOMATIC management policy, [3-8](#)

Automatic Performance Diagnostics (AWR)
 monitor performance statistics, [14-6](#)

automatic segment space management (ASSM),
[13-6](#)
 tablespace use in Oracle RAC, [13-6](#)

automatic undo management
 tablespace use in Oracle RAC, [13-6](#)

automatic workload management, [1-21](#)
 concepts, [1-21](#), [5-24](#)
 manual rebalancing, [5-20](#)

Automatic Workload Repository (AWR), [1-21](#),
[5-20](#), [5-21](#), [14-6](#), [14-8](#), [14-9](#)
 monitoring performance, [5-25](#)
 snapshots, [14-6](#)

AVAILABLE instances
 for services, [5-27](#)

Average Active Sessions chart
 performance monitoring, [14-3](#)

AWR
 see Automatic Workload Repository (AWR),
[14-9](#)

B

background processes
 Oracle RAC, [1-20](#)
 SMON, [3-9](#), [8-3](#)

background thread trace files, [B-1](#)

backups
 and converting to cluster database, [15-1](#)
 server parameter file, [3-17](#)

bandwidth
 interconnect, [14-4](#)

best practices
 deploying Oracle RAC for high availability,
[13-2](#)

blackouts
 defining, [3-32](#)

block mode conversions
 statistics for, [14-6](#)

blocks
 associated with instance, [8-3](#)

buffer cache, [1-19](#)
 instance recovery, [8-3](#)

buffer sizes
 interprocess communication (IPC)
 adjusting for Oracle RAC, [14-5](#)

C

cache coherency, [14-9](#)

Cache Fusion, [1-19](#), [13-7](#)
 and e-commerce applications, [13-7](#)
 overview, [1-35](#)
 performance, [14-4](#)

- Cache Fusion (*continued*)
 - transfers, [14-10](#)
- callouts
 - how they are run, [6-9](#)
- capacity
 - increasing, [13-5](#)
- cardinality, [3-1](#)
- catclustdb.sql script, [1-34](#)
- CDBs, [1-32](#), [3-11](#)
- changing the configuration of all services, [4-2](#)
- channels
 - configure one RMAN channel for each Oracle RAC instance, [7-4](#)
 - configuring during crosscheck or restore operations, [7-3](#)
 - configuring for RMAN, [7-3](#)
- charts
 - Average Active Sessions, [14-3](#)
 - Cluster Host Load Average, [14-3](#)
 - Database Throughput, [14-3](#)
 - Global Cache Block Access Latency, [14-3](#)
- checking the interconnect, [B-4](#)
- client connections with SCAN, [3-1](#)
- client drivers
 - and FAN, [5-8](#)
- client-side load balancing, [5-1](#), [5-4](#)
- clients
 - application environments and FAN, [5-8](#)
 - integrated for FAN events, [6-2](#)
 - JDBC-thin driver, [5-9](#)
 - JDBC/OCI, [5-10](#)
- clone.pl, [9-6](#)
- clone.pl script
 - cloning parameters, [9-3](#)
 - environment variables, [9-3](#)
- cloning, [1-6](#), [9-1](#)
 - deployment phase, [9-3](#)
 - log files, [9-6](#)
 - parameters passed to the clone.pl script, [9-3](#)
 - preparation phase, [9-2](#)
 - running \$ORACLE_HOME/root.sh script, [9-3](#)
- cluster
 - definition of, [1-1](#)
- cluster administrator, [3-1](#)
- cluster cache coherency, [14-3](#)
- cluster configuration policy, [1-29](#)
- cluster configuration policy set, [1-29](#)
- Cluster Database Performance page
 - Top Activity drill down menu, [14-3](#)
- cluster databases
 - creating using DBCA, [15-5](#)
- cluster file system
 - archiving parameter settings, [7-7](#)
 - archiving scenario, [7-6](#)
 - restore, [8-2](#)
- cluster file system (*continued*)
 - storage in Oracle RAC, [2-2](#)
- Cluster Host Load Average page
 - cluster database performance, [14-3](#)
- cluster nodes name
 - in clone.pl script, [9-3](#)
- Cluster Verification Utility, [1-33](#)
 - overview and concepts, [1-33](#)
- CLUSTER_DATABASE initialization parameter, [3-18](#), [3-20](#)
- CLUSTER_DATABASE_INSTANCES
 - initialization parameter, [3-18](#), [3-22](#)
- CLUSTER_INTERCONNECTS
 - parameter, [14-5](#)
- CLUSTER_INTERCONNECTS initialization parameter, [3-18](#), [3-26](#)
- clustered Oracle ASM
 - converting a noncluster Oracle ASM, [2-6](#)
- clusters
 - consolidating multiple databases in, [13-3](#)
 - policy-managed, [1-29](#)
- clusterware management solution, [1-8](#)
- cold backups, [3-25](#)
- comma-delimited lists with SRVCTL, [A-1](#)
- command-line interpreter
 - ADR Command-Line Interpreter (ADRCI), [B-2](#)
- committed data
 - instance failure, [8-3](#)
- communication protocols
 - verifying settings for, [14-4](#)
- compatibility
 - Oracle RAC and Oracle Database software, [1-4](#)
- COMPATIBLE initialization parameter, [3-20](#)
- configuring channels
 - during restore or crosscheck operations, [7-3](#)
- configuring preferred mirror read disks in
 - extended clusters, [2-6](#)
- CONNECT command, [3-6](#)
 - SQL*Plus, [3-6](#)
- CONNECT SYS
 - example of, [3-10](#)
- connecting
 - to instances, [1-33](#)
- connection load balancing
 - introduction to, [1-21](#)
 - long method, [5-2](#)
 - short method, [5-2](#)
- connection pools
 - and FAN, [5-8](#)
- connection tests, [6-16](#)
 - adding, [6-16](#)
 - disabling, [6-16](#)
 - enabling, [6-16](#)

connection tests (*continued*)
 removing, [6-16](#)
 consistent blocks, [1-19](#)
 container databases
 See CDBs
 CONTROL_FILES initialization parameter, [3-20](#)
 convert to cluster database
 from single-instance to Oracle Real
 Application Clusters, [15-1](#)
 post-conversion, [15-14](#)
 to Oracle RAC from single-instance
 databases, [15-1](#)
 convert to Oracle RAC database
 from non-cluster system, [15-2](#)
 converting
 from single instance to Oracle RAC, [15-7](#)
 from single-instance to Oracle Real
 Application Clusters, [15-10](#)
 with rconfig, [15-7](#)
 converting a database from Oracle RAC One
 Node to Oracle RAC, [4-3](#)
 converting an Oracle RAC database with one
 instance to Oracle RAC One Node, [4-2](#)
 corrupted data blocks, [B-2](#)
 CREATE PFILE
 FROM MEMORY clause, [3-16](#)
 CREATE PFILE statement, [3-17](#)
 CREATE SPFILE
 FROM MEMORY clause, [3-16](#)
 creating
 server parameter files, [3-15](#)
 services, [5-31](#)
 SPFILE backups, [3-17](#)
 crosscheck operations
 configuring channels during, [7-3](#)
 crosschecking on multiple nodes
 RMAN backups, [7-3](#)
 CRS resources
 management of, [1-8](#)
 current blocks, [1-19](#)
 CVU
 See Cluster Verification Utility

D

data dictionary
 querying views, [14-6](#)
 Data Recovery Advisor, [B-2](#)
 data security
 keystore, [13-8](#)
 data warehouse
 deploying applications for in Oracle RAC,
 [13-8](#)
 data warehouse systems, [13-8](#)
 data-dependent routing, [5-19](#)

database
 administrative privilege
 SYSDBA, [3-1](#)
 draining, [6-16](#)
 services
 singleton, [3-23](#)
 uniform, [3-23](#)
 SRVCTL object name, [A-15](#)
 database cloud, [13-4](#)
 Database Configuration Assistant (DBCA)
 adding and deleting instances in interactive
 mode
 on Windows, [12-4](#)
 adding and deleting instances in silent mode
 on Windows, [12-5](#)
 adding instances in interactive mode
 on Linux and UNIX, [11-4](#)
 adding instances in silent mode
 on Linux and UNIX, [11-5](#)
 cloning Oracle RAC instances, [9-3](#)
 creating views for Oracle Real Application
 Clusters, [14-6](#)
 Database Storage page, [11-4](#), [12-4](#)
 deleting instances in interactive mode
 on Linux and UNIX, [11-6](#)
 on Windows, [12-6](#)
 deleting instances in silent mode
 on Linux and UNIX, [11-7](#)
 on Windows, [12-7](#)
 Instance Management page, [11-4](#), [12-4](#)
 List of Cluster Databases page, [11-4](#), [12-4](#)
 running the catclustdb.sql script, [1-34](#)
 Welcome page, [11-4](#)
 database deployment
 administrator-managed, [1-24](#), [3-1](#)
 policy-managed, [1-24](#), [3-1](#)
 database instances
 administrator managed
 deleting, [11-6](#), [12-6](#)
 connecting to, [3-5](#)
 database pools, [13-4](#)
 database resource, [3-1](#)
 Database Resource Manager, [3-25](#)
 database role, [3-8](#)
 Database Storage page, [11-4](#), [12-4](#)
 Database Throughput page
 performance monitoring, [14-3](#)
 databases
 administrator managed, [5-27](#)
 consolidating multiple in a cluster, [13-3](#)
 controlling restarts, [3-28](#)
 creating
 Oracle RAC One Node, [4-1](#)
 Oracle RAC One Node
 services on, [4-1](#)

databases (*continued*)
 policy managed, [3-1](#), [5-28](#)
 scalability, [13-5](#)

databases sessions
 pre-created, [5-13](#)

DB_BLOCK_SIZE initialization parameter, [3-20](#)

DB_DOMAIN initialization parameter, [3-21](#)

DB_FILES initialization parameter, [3-21](#)

DB_NAME initialization parameter, [3-18](#), [3-21](#)

DB_RECOVERY_FILE_DEST initialization parameter, [3-21](#), [8-7](#)

DB_RECOVERY_FILE_DEST_SIZE initialization parameter, [3-21](#)

DB_UNIQUE_NAME initialization parameter, [3-21](#)

DDL statements, [13-6](#)

default database service, [1-5](#), [3-18](#), [5-31](#)

degree of parallelism, [13-8](#)

deleting administrator-managed database instances, [11-6](#), [12-6](#)

dependencies
 and services, [5-25](#)

deploying
 Oracle Real Application Clusters environments, [1-33](#), [13-1](#)

design
 Oracle Real Application Clusters environments, [1-33](#), [13-1](#)

diagnosing problems for Oracle RAC, [B-1](#)

diagnosing problems using ADR, [B-2](#)

DISPATCHERS initialization parameter, [3-18](#)
 specifying a service with, [5-31](#)

Distributed Transaction Processing (DTP), [5-16](#), [13-7](#)
See also DTP

distributed transactions, [13-7](#)
 directing to a single instance in the cluster, [5-17](#)
 services in Oracle RAC, [5-16](#)
 XA transactions span instances, [5-16](#)

DML_LOCKS initialization parameter, [3-21](#)

draining database sessions, [6-16](#)

DTP, [5-16](#)
 services
 with Oracle RAC, [5-18](#)
 XA affinity, [5-17](#)
 XA transactions, [5-16](#)

DTP service., [5-17](#)

dynamic database services
 description, [1-21](#)

Dynamic Database Services
 introduction to, [1-18](#)

dynamic performance views, [14-8](#)
 creating, [14-6](#)
 GV\$, [1-34](#)

dynamic performance views (*continued*)
 V\$, [1-34](#)

dynamic resource allocation
 overview, [1-35](#)

dynamic session state consistency, [6-46](#)

E

e-commerce
 applications in Oracle RAC, [13-7](#)

edition
 services attribute, [5-26](#)

ENCRYPTION_WALLET_LOCATION parameter, [13-8](#)

Enterprise Manager
 overview, [1-8](#)

environment variables
 passed to the clone.pl script, [9-3](#)
 setting with SRVCTL, [3-4](#)

evaluating block transfers, [14-6](#)

event notification
 enabling, [5-13](#)

extended distance cluster, [2-6](#)
 Oracle ASM preferred mirror read, [2-6](#)

extended distance clusters
 configuring preferred mirror read disks, [2-6](#)

extending Oracle database home
 on shared storage
 network-attached storage, [11-1](#), [12-1](#)
 Oracle ACFs, [11-1](#), [12-1](#)

external transaction managers
 OraMTS, [5-16](#)

F

failure
 instance, [8-3](#)
 multiple node, [8-4](#)
 node, [8-3](#)

failure groups, [2-6](#)

FAN
See Fast Application Notification (FAN)

Fast Application Notification (FAN), [5-10](#), [6-2](#)
 and high availability events, [6-6](#)

callouts
 definition, [6-8](#)
 how to use, [6-8](#)

events
 enabling for JDBC, [5-10](#)
 enabling for JDBC-thin clients, [5-9](#)
 enabling for OCI, [5-12](#)
 enabling for ODP.NET, [5-14](#)
 enabling ODP.NET clients, [5-15](#)

HA events, [5-8](#)
 how events are published, [6-2](#)

Fast Application Notification (FAN) (*continued*)
 introduction, [1-21](#)
 overview, [6-2](#)
 parameters and matching database signatures, [6-6](#)
 uses, [6-2](#)

Fast Connection Failover (FCF), [5-9](#)
 enabling JDBC-thin clients, [5-9](#)
 enabling with thin and thick clients, [5-10](#)
 introduction to, [1-21](#)

Fast Recovery
 and files managed by Oracle, [15-10](#)

fault diagnosability, [B-2](#)

FCF
 See Fast Connection Failover (FCF)

files
 archived redo log files, [7-5](#)
 redo log, [7-5](#)

FROM MEMORY clause
 on CREATE PFILE or CREATE SPFILE, [3-16](#)

G

GC_SERVER_PROCESSES initialization parameter
 specifying the number of LMSn processes, [13-4](#)

GCS protocol, [14-9](#)

GCS_SERVER_PROCESSES initialization parameter, [3-18](#)

Generic server pool, [3-1](#)

GES
 See global cache and enqueue service (GES)

global cache and enqueue service (GES), [14-8](#)

Global Cache Block Access Latency chart
 performance monitoring, [14-3](#)

Global Cache Service (GCS), [1-19](#), [3-18](#)

Global Cache Service Process (LMS), [1-20](#)

Global Cache Service Processes (LMSn)
 reducing the number of, [13-4](#)
 specifying the number of, [13-4](#)

Global Cache Service statistics, [14-8](#), [14-9](#)

GLOBAL clause
 forcing a checkpoint, [3-6](#)

Global Enqueue Service (GES), [1-19](#)

Global Enqueue Service Daemon (LMD), [1-20](#)

Global Enqueue Service Monitor (LMON), [1-20](#)

Global Enqueue Service statistics, [14-8](#)

global performance data
 with ADDM, [14-6](#)

Global Resource Directory (GRD), [1-19](#)

global service attributes
 GDSCTL, [5-37](#)

global services, [5-37](#), [13-4](#)

Global Transaction Process (GTX0-j), [1-20](#)

GLOBAL_TXN_PROCESSES initialization parameter, [5-16](#)

goals
 and the load balancing advisory, [5-5](#)
 for load balancing advisory, [5-5](#)

GTX0-j
 Global Transaction Process, [1-20](#)

GV\$, [1-34](#)

GV\$ view, [14-5](#)

GV\$SESSION, [3-13](#)

H

Hang Manager, [1-31](#)

hash partitioning
 with Oracle RAC, [13-6](#)

high availability
 best practices, [13-2](#)
 for Oracle RAC Databases, [13-1](#)

high availability framework
 introduction to, [1-21](#)

HOST command, [3-6](#)
 SQL*Plus, [3-6](#)

I

idempotence, [6-53](#)

idle wait class, [14-7](#)

IM column store
 See In-Memory Column Store

In-Memory Column Store,
 and Oracle RAC, [1-32](#)
 overview, [1-32](#)

In-Memory FastStart, [1-32](#)

indexes
 sequence-based, [13-6](#)

initialization parameters
 cluster database issues regarding, [3-18](#)
 CLUSTER_INTERCONNECTS, [3-26](#), [14-5](#)
 recommendations for using, [3-26](#)
 identical settings for on all instances, [3-22](#)
 RECOVERY_PARALLELISM, [8-7](#)
 settings for instances, [3-16](#)
 specific to Oracle RAC, [3-18](#)
 that must be identical on all instances, [3-20](#)
 that must be unique on all instances, [3-21](#)

INST_ID column, [14-5](#)

installations
 performing multiple simultaneous cluster, [9-1](#)

instance
 SRVCTL object name, [A-15](#)

instance discovery
 Oracle Enterprise Manager Cloud Control, [3-29](#)

Instance Enqueue Process (LCK0), [1-20](#)
 Instance Management page, [11-4](#), [12-4](#)
 INSTANCE_NAME initialization parameter, [3-18](#)
 INSTANCE option, [3-6](#)
 instance recovery, [3-9](#)
 INSTANCE_NAME initialization parameter, [3-21](#)
 INSTANCE_NUMBER initialization parameter, [3-21](#)
 INSTANCE_TYPE initialization parameter, [3-21](#)
 instances
 aggregated for service performance, [14-3](#)
 cloning Oracle RAC, [9-3](#)
 effect of SQL*Plus commands on, [3-6](#)
 failure, [8-4](#)
 initialization parameter settings, [3-16](#)
 maximum number for Oracle RAC, [1-1](#)
 memory structures, [1-19](#)
 parallel processes, [5-25](#)
 private interconnect usage, [B-3](#)
 recovery, [3-9](#), [8-3](#)
 recovery, abnormal shutdown, [3-9](#)
 recovery, multiple failures, [8-4](#)
 Server Management, [3-4](#)
 shutting down, [3-9](#)
 starting and stopping, [3-7](#)
 terminating session on, [3-13](#)
 verifying, [3-12](#)
 verifying running, [3-12](#)
 instances failure
 recovery from, [8-3](#)
 instead of Statspack, [1-34](#)
 interconnect
 and performance, [14-5](#)
 and the Oracle RAC architecture, [1-1](#)
 defined, [1-17](#)
 protocols for Oracle RAC, [14-4](#)
 verifying settings for, [14-4](#)
 interconnect bandwidth, [14-4](#)
 latency, [14-4](#)
 interconnect block transfer rates, [14-6](#)
 interconnect settings
 verifying, [14-4](#)
 interconnects
 alternatives to the private network, [3-26](#)
 private, [B-3](#)
 Interconnects page
 monitoring clusterware with Oracle Enterprise Manager, [14-1](#)
 monitoring Oracle Clusterware, [14-2](#)
 interprocess communication (IPC)
 buffer sizes
 adjusting, [14-5](#)
 IPC protocol, [14-4](#), [14-9](#)

J

Java Database Connectivity (JDBC) clients
 Oracle Notification Service usage, [1-21](#)
 Java-based tools and utilities
 CVU, [B-3](#)
 DBCA, [B-3](#)
 DBUA, [B-3](#)
 enabling tools for tracing, [B-3](#)
 GSD, [B-3](#)
 NETCA, [B-3](#)
 SRVCTL, [B-3](#)
 JDBC
 enabling Fast Application Notification events for, [5-10](#)
 JDBC-thin clients
 enabling for Fast Connection Failover (FCF), [5-9](#)
 JDBC-thin driver, [5-9](#)
 JDBC/OCI, [5-10](#)
 job administration
 Oracle Enterprise Manager, [3-31](#)

K

keystore
 data security, [13-8](#)
 KILL SESSION clause
 on the ALTER SYSTEM statement, [3-13](#)

L

LCK0
 Instance Enqueue Process, [1-20](#)
 level thresholds
 services, [5-22](#)
 LICENSE_MAX_USERS initialization parameter, [3-22](#)
 List of Cluster Databases page, [11-4](#), [12-4](#)
 listener
 SRVCTL object name, [A-15](#)
 listeners
 command to add to a node, [A-48](#)
 command to remove, [A-54](#)
 Oracle Net, [1-8](#)
 LMD
 Global Enqueue Service Daemon, [1-20](#)
 LMON Global Enqueue Service Monitor, [1-20](#)
 LMS
 Global Cache Service Process, [1-20](#)
 LMS processes
 reducing the number of, [13-4](#)
 LMSn processes
 reducing the number of, [13-4](#)
 load balancing, [13-7](#)

load balancing (*continued*)
 OCI runtime connection, [5-13](#)
 server-side, [5-1](#)

load balancing advisory
 and FAN events, [5-6](#)
 configuring your environment for using, [5-5](#)
 deployment, [5-5](#)
 description of, [5-5](#)
 events and FAN, [6-2](#)
 introduction to, [1-21](#)

Load Balancing Advisory, [5-29](#)

local archiving scenario
 RMAN, [7-7](#)

Local Area Network (LAN), [1-17](#)

LOCAL clause
 forcing a checkpoint, [3-6](#)

local file system
 archiving parameter settings, [7-8](#)
 restore, [8-2](#)

local instance
 shutting down, [3-10](#)
 starting, [3-10](#)

local node name
 in clone.pl script, [9-3](#)

local temporary tablespace, [1-10](#)

LOCAL_NODE parameter
 in clone.pl script, [9-3](#)

locally managed tablespaces, [13-6](#)

log files
 tracing, [B-3](#)

log sequence numbers, [7-5](#)

LOG_ARCHIVE_FORMAT initialization
 parameter, [3-22](#)

LOG_ARCHIVE_FORMAT parameter, [7-5](#)

logical transaction ID, [6-53](#)

LXTID, [6-53](#)

M

maintenance
 planned, [6-10](#)

mass deployment
 cloning, [9-1](#), [9-2](#)

media failures
 recovery from, [8-6](#)

Memory Guard, [3-24](#)

memory structures
 in Oracle RAC, [1-19](#)

message request counters, [14-6](#)

migration
 application, [13-5](#)
 from single-instance, [15-2](#)

missing files, [B-2](#)

mission critical systems
 considerations for Oracle RAC, [13-1](#)

modified data
 instance recovery, [8-3](#)

monitoring
 archiver process, [7-10](#)
 overview and concepts, [1-34](#)
 performance of global cache block access,
[14-3](#)

monitoring host load average, [14-3](#)

mount all non-running instances of an Oracle
 RAC database, [3-8](#)

multiple cluster interconnects, [3-26](#)

multiple databases in a cluster, [3-27](#)

multiple instances
 starting from a single SQL*Plus session, [3-10](#)
 starting from one node, [3-10](#)

multiple node failures, [8-4](#)

multiple public networks, [1-19](#)

multiplexed redo log files, [2-3](#)

multitenant container database
 See CDBs

mutable functions, [6-43](#)

mutables
 rules for grants, [6-45](#)

N

net service name, [3-5](#)

network
 restricting service registration, [1-19](#), [5-30](#)
 SRVCTL object name, [A-15](#)

Network Attached Storage (NAS), [1-17](#)

network file system, [2-3](#)
 See also NFS

network resources, [1-19](#)

NFS, [2-3](#)
 server, [2-3](#)

node
 failure and VIP addresses, [1-18](#)

node affinity awareness, [8-5](#)

node discovery
 Oracle Enterprise Manager Cloud Control,
[3-29](#)

node evictions, [14-2](#)

node VIPs, [1-18](#)

nodeapps
 SRVCTL object name, [A-15](#)

nodes
 affinity awareness, [8-5](#)
 failure of, [8-3](#)
 virtual IP addresses, [A-4](#)

non-transactional session state, [6-49](#)

noncluster databases
 quiescing, [3-25](#)

noncluster Oracle ASM
 converting to clustered Oracle ASM, [2-6](#)

noreplay keyword (terminating or disconnecting session), [6-42](#)

O

object creation and deletion, [13-6](#)

objects

- creation of and effect on performance, [13-6](#)
- srvctl object names and abbreviations, [A-15](#)

OCI, [5-13](#)

- runtime connection load balancing, [5-13](#)
- session pooling, [5-13](#)
- session pools
 - optimizing, [5-13](#)
 - runtime connection load balancing, [5-13](#)
 - service metrics, [5-13](#)

OCRDump utility, [14-5](#)

ODP.NET

- and Fast Connection Failover, [5-14](#)
- load balancing advisory events, [5-15](#)

OLTP environments, [13-6](#)

online database relocation

- relocation feature, [4-3](#)

online recovery, [8-3](#)

online transaction processing (OLTP)

- applications in Oracle RAC, [13-7](#)

ons

- SRVCTL object name, [A-15](#)

ONS

- See Oracle Notification Service

optimal execution plans, [13-8](#)

Oracle ACFS, [1-17](#)

Oracle ASM

- disk group management, [2-6](#)
- see Oracle Automatic Storage Management (Oracle ASM), [2-2](#)

Oracle Automatic Storage Management (Oracle ASM), [2-1](#)

- archiving scenario, [7-6](#)
- converting noncluster Oracle ASM to clustered Oracle ASM, [2-6](#)
- installation, [1-5](#)
- instances
 - administering with SRVCTL, [2-6](#)
- Oracle ASM preferred read failure groups, [2-6](#)
- preferred mirror read disks, [2-6](#)
- preferred read disks, [3-18](#)
- storage solution, [2-2](#)

Oracle Call Interface

- See OCI

Oracle Cluster Registry (OCR), [3-4](#), [14-5](#)

Oracle Clusterware

- cloning, [1-6](#)
- control policies

Oracle Clusterware (*continued*)

control policies (*continued*)

AUTOMATIC, [3-28](#)

MANUAL, [3-28](#)

using SRVCTL to display and change, [3-28](#)

controlling database restarts, [3-28](#)

described, [1-8](#)

introduction, [1-8](#)

introduction and concepts of, [1-1](#)

managing Oracle processes, [1-8](#)

Oracle Database

session activity, [14-7](#)

Oracle Database QoS Management, [1-30](#)

Oracle Database Quality of Service Management

See Oracle Database QoS Management

Oracle Database Upgrade Assistant, [15-1](#)

Oracle Enterprise Manager

adding database instances to nodes

on Linux and UNIX, [11-3](#)

on Windows, [12-3](#)

alert administration, [3-31](#)

alert blackouts, [3-32](#)

Automatic Database Diagnostic Monitor

(ADDM), [14-6](#)

Average Active Sessions chart, [14-3](#)

Cluster Database Home page, [14-2](#)

Cluster Database page, [14-1](#)

Cluster Database Performance page

performance statistics for an Oracle RAC database, [14-3](#)

configuring to recognize changes in database

management, [3-23](#)

Database Throughput charts, [14-3](#)

deleting database instances from nodes,

[11-6](#), [12-6](#)

Global Cache Block Access Latency chart,

[14-3](#)

Interconnects page, [14-2](#)

job administration, [3-31](#)

monitoring load values for available nodes,

[14-3](#)

overview and concepts, [1-33](#)

Top Activity drill down menu, [14-3](#)

using the Interconnects page to monitor

Oracle Clusterware, [14-1](#)

using to administer Oracle RAC, [3-4](#)

using to administer services, [5-32](#)

using to back up the server parameter file,

[3-17](#)

using to convert single-instance databases to

Oracle Real Application Clusters,

[15-10](#)

using to create DTP services, [5-18](#)

using to monitor Oracle Clusterware, [14-1](#)

- Oracle Enterprise Manager (*continued*)
 - using to monitor Oracle RAC, [14-1](#)
 - using to monitor Oracle RAC environments, [1-34](#)
 - using to restore SPFILE, [8-3](#)
 - using to schedule Automatic Workload Repository actions, [5-22](#)
 - using to set up a fast recovery area, [8-7](#)
 - using to start or stop a database, [3-7](#)
 - using with RMAN, [7-2](#)
- Oracle Enterprise Manager Cloud Control
 - instance discovery, [3-29](#)
 - node discovery, [3-29](#)
- Oracle Flex Clusters, [1-9](#)
- Oracle Grid Infrastructure, [1-1](#)
- Oracle home
 - defined, [1-4](#)
- Oracle homes
 - cloning
 - on Linux and UNIX, [10-1](#)
 - local
 - cloning on Windows, [10-3](#)
 - shared
 - cloning on Linux and UNIX, [10-2](#)
 - cloning on Windows, [10-3](#)
- Oracle Interface Configuration (OIFCFG), [14-5](#)
- Oracle Managed Files, [3-1](#)
- Oracle Maximum Availability Architecture (MAA), [13-1](#)
- Oracle Multitenant, [1-32](#)
- Oracle Net
 - listeners, [1-8](#)
- Oracle Net connection failover, [5-2](#)
- Oracle Net Services
 - and load balancing, [5-8](#)
 - and services, [5-29](#)
- Oracle Notification Service, [1-8](#)
 - SRVCTL object name, [A-15](#)
 - used by FAN, [1-21](#)
- Oracle Notification Services
 - API, [6-2](#)
- Oracle processes
 - managed by Oracle Clusterware, [1-8](#)
- Oracle RAC, [1-1](#)
 - adding administrator-managed database instances, [11-4](#), [12-3](#)
 - adding policy-managed database instances, [11-1](#), [12-1](#)
 - adding to nodes in a cluster on Linux and UNIX, [11-1](#)
 - adding to nodes in a cluster on Windows, [12-1](#)
 - administrative privilege
 - SYSRAC, [3-1](#)
 - and e-commerce, [13-7](#)
- Oracle RAC (*continued*)
 - background processes
 - ACMS, [1-20](#)
 - GTX0-j, [1-20](#)
 - LCK0, [1-20](#)
 - LMD, [1-20](#)
 - LMON, [1-20](#)
 - LMS, [1-20](#)
 - RSMN, [1-20](#)
 - benefits of cloning, [9-1](#)
 - cloning, [9-1](#)
 - size of the binaries, [9-2](#)
 - converting database from, [4-2](#)
 - converting database to, [4-2](#)
 - converting from non-cluster system, [15-2](#)
 - converting single instance to Oracle RAC One Node, [4-2](#)
 - copying the Oracle RAC home, [9-2](#)
 - databases
 - converting from single-instance Oracle databases, [15-1](#)
 - quiescing, [3-25](#)
 - deploying clone, [9-3](#)
 - diagnosing performance problems, [14-7](#)
 - diagnosing problems for, [B-1](#)
 - IM column store, [1-32](#)
 - installation overview, [1-3](#)
 - overview of administration, [1-1](#)
 - removing on Windows, [12-8](#)
 - removing the software from Linux and UNIX, [11-8](#)
 - security considerations, [13-8](#)
 - shutting down instances, [3-9](#)
 - software components, [1-19](#)
 - storage options
 - IBM GPFS, [1-17](#)
 - network file system (NFS), [1-17](#)
 - Oracle Automatic Storage Management (Oracle ASM), [1-17](#)
 - Oracle Automatic Storage Management Cluster File System (Oracle ACFS), [1-17](#)
 - Oracle OCFS2, [1-17](#)
 - volume manager, [1-17](#)
 - using a fast recovery area in, [8-7](#)
- Oracle RAC One Node, [4-1](#)
 - converting database from, [4-2](#)
 - converting database to, [4-2](#)
 - converting to Oracle RAC, [4-3](#)
 - creating databases, [4-1](#)
 - database
 - services on, [4-1](#)
 - online database relocation, [4-3](#)
 - relocating to another node, [4-3](#)
- Oracle RAC Sharding, [5-19](#)

Oracle Real Application Clusters
 See Oracle RAC

Oracle Real Application Clusters One Node
 See Oracle RAC One Node

Oracle Resource Manager
 and services, [5-25](#)

Oracle Services
 using, [5-30](#)

Oracle Services for Microsoft Transaction Server,
[12-4](#)
 creating the OraMTS service, [12-4](#)

Oracle Streams, [5-25](#)

Oracle Streams Advanced Queuing
 and FAN, [6-2](#)

Oracle Universal Installer
 database installation, [1-4](#)
 Oracle Real Application Clusters installation,
[1-4](#)

Oracle XA, [5-29](#)

oradebug ipc command, [B-4](#)

OraMTS, [12-4](#)
 external transaction manager, [5-16](#)
 See also Oracle Services for Microsoft
 Transaction Server

orapwd file, [3-23](#)

outages
 unplanned, [6-9](#)

P

parallel execution, [13-8](#)

parallel processes, [5-25](#)

parallel recovery, [8-7](#)
 disabling, [8-7](#)

PARALLEL_EXECUTION_MESSAGE_SIZE
 initialization parameter, [3-21](#)

PARALLEL_FORCE_LOCAL initialization
 parameter, [5-25](#)

parallelism
 in Oracle RAC, [13-8](#)
 parallel-aware query optimization, [13-8](#)

parameter file
 overview, [3-15](#)

parameter file search order, [3-17](#)

parameters
 DB_RECOVERY_FILE_DEST, [8-7](#)
 that must be identical on all instances, [3-20](#)
 that must be unique on all instances, [3-21](#)

password file-based authentication, [4-3](#)

PDBs, [1-32](#), [3-11](#)
 managing, [3-11](#)
 managing services on, [6-14](#)

performance, [14-3](#)
 aggregates by services, [14-3](#)
 comprehensive global data, [14-6](#)

performance (*continued*)
 monitoring activity by wait events, services,
 and instances, [14-3](#)
 monitoring database throughput, [14-3](#)
 monitoring global cache block access, [14-3](#)
 monitoring potential problems in the
 database, [14-3](#)
 primary components affecting, [14-4](#)
 service aggregates by instances, [14-3](#)
 service aggregates by waits, [14-3](#)
 using ADDM, [1-34](#)

performance evaluation
 overview and concepts, [1-35](#)

performance statistics, [14-11](#)

PFILE
 using with Oracle RAC, [3-15](#)

phases
 cloning deployment, [9-3](#)
 cloning preparation, [9-2](#)

planned maintenance
 draining database sessions, [6-16](#)

planned maintenance
 managing, [6-11](#)
 using Application Continuity, [6-39](#)

pluggable database
 See PDBs

PMON background process, [3-13](#)

policy set, [1-29](#)

policy-managed clusters, [1-29](#)

policy-managed database, [3-1](#)

policy-managed database instances
 adding, [12-1](#)

policy-managed databases, [1-24](#), [3-1](#), [3-21](#), [5-28](#)
 deleting on Linux and UNIX, [11-6](#)
 deleting on Windows, [12-6](#)
 deploying, [1-28](#)
 managing, [1-28](#)

PREFERRED instances
 for services, [5-27](#)

preferred read disks
 Oracle ASM in an Oracle RAC extended
 distance cluster, [2-6](#)

private clouds, [13-4](#)

private interconnect, [B-4](#)
 determining usage, [B-3](#)

private network
 alternative interconnect, [3-26](#)

private network IP address, [14-5](#)

processes
 managed by Oracle Clusterware, [1-8](#)
 parallel, [5-25](#)

public and private interfaces
 shown in Oracle Enterprise Manager, [14-2](#)

public network
 defined, [1-17](#)

Q

- query optimizer, [13-8](#)
 - default cost model for, [13-8](#)
- queue tables, [5-25](#)
- quiesce database
 - in Oracle Real Application Clusters, [3-25](#)
- quiesced databases, [3-25](#)
- quiescing
 - noncluster database, [3-25](#)

R

- rconfig, [15-7](#)
- reader nodes, [1-9](#)
- rebalancing
 - workloads, [5-20](#)
- RECOVER command, [3-6](#)
 - SQL*Plus, [3-6](#)
- recovery
 - after SHUTDOWN ABORT, [3-9](#)
 - from multiple node failure, [8-4](#)
 - from single-node failure, [8-3](#)
 - instance, [3-9](#)
 - media failures, [8-6](#)
 - online, [8-3](#)
 - parallel, [8-7](#)
- RECOVERY_PARALLELISM parameter, [8-7](#)
- redo log files
 - instance recovery, [8-3](#)
 - log sequence number, [7-5](#)
 - using, [2-3](#)
- redo log groups, [2-3](#)
- redo logs
 - format and destination specifications, [7-5](#)
- redo thread, [3-1](#)
- reducing contention, [13-6](#)
- regions, [13-4](#)
- remote Oracle Notification Service subscription, [5-9](#)
- Remote Slave Monitor (RSMN), [1-20](#)
- REMOTE_LOGIN_PASSWORDFILE initialization parameter, [3-21](#)
- replicated databases
 - and global services, [5-37](#)
- request boundaries, [6-23](#), [6-24](#)
- resource contention, [14-3](#)
- resource manager, [5-20](#)
- resource profiles
 - and service creation, [5-25](#)
- resources
 - releasing, [8-3](#)
- resources held by the session, [3-13](#)
- restore scenarios
 - RMAN, [8-1](#)

- restore scheme
 - cluster file system, [8-2](#)
 - local file system, [8-2](#)
- result cache, [3-18](#)
 - disabling, [3-18](#)
 - enabling, [3-18](#)
- RESULT_CACHE_MAX_SIZE initialization parameter, [3-18](#), [3-21](#)
- RMAN
 - CONFIGURE command, [7-2](#)
 - configuring channels, [7-3](#)
 - configuring channels to use automatic load balancing, [7-3](#)
 - configuring one channel for each instance, [7-4](#)
 - configuring snapshot control file locations, [7-2](#)
 - crosschecking on multiple nodes, [7-3](#)
 - local archiving scenario, [7-7](#)
 - restore scenarios, [8-1](#)
 - using to create SPFILE backups, [3-17](#)
- RMSn
 - Oracle RAC management processes, [1-20](#)
- rolling back
 - instance recovery, [8-3](#)
- root.sh script
 - \$ORACLE_HOME, [9-3](#)
- RSMN
 - Remote Slave Monitor, [1-20](#)
- runtime connection load balancing
 - defined, [5-13](#)
 - in OCI session pools, [5-13](#)
 - introduction to, [1-21](#)
- Runtime Connection Load Balancing, [5-28](#)

S

- scalability, [13-7](#)
 - Oracle RAC, [13-5](#)
- scan
 - SRVCTL object name, [A-15](#)
- SCAN, [1-8](#)
- SCAN listener
 - and restricting service registration, [1-19](#), [5-30](#)
- scan_listener
 - SRVCTL object name, [A-15](#)
- scripts
 - \$ORACLE_HOME/root.sh, [9-3](#)
- security
 - keystore, [13-8](#)
- sequence-based indexes, [13-6](#)
- sequences
 - log sequence number, [7-5](#)
- Server Control Utility
 - See SRVCTL

- server draining, [6-16](#)
- Server Management
 - administration of instances, [3-4](#)
- server parameter file
 - backing up, [3-17](#)
- server parameter files
 - creating, [3-15](#)
- server pool, [3-1](#)
- server pools, [1-24](#)
 - creating for a policy-managed database, [3-23](#)
 - Generic, [3-1](#)
 - XML conversion files for, [15-12](#)
- servers
 - relocating from another server pool, [3-23](#)
 - scalability, [13-5](#)
- service
 - SRVCTL object name, [A-15](#)
- service level objective
 - defining for Oracle RAC, [13-1](#)
- service levels, [13-7](#)
- service metrics
 - OCI runtime connection load balancing, [5-13](#)
 - runtime connection load balancing, [5-13](#)
- SERVICE TIME
 - load balancing advisory goal, [5-5](#)
- SERVICE_NAMES initialization parameter, [3-18](#)
 - setting for services, [3-18](#), [5-31](#)
- service-level agreements, [3-13](#)
- services, [3-18](#)
 - activity levels aggregated by instances, [14-3](#)
 - activity levels aggregated by services, [14-3](#)
 - activity levels aggregated by waits, [14-3](#)
 - administering, [5-31](#)
 - administering with Oracle Enterprise Manager, [5-30](#)
 - administering with SRVCTL, [5-30](#), [5-33](#)
 - attributes
 - edition, [5-26](#)
 - basic concepts about, [5-24](#)
 - buffer cache access, [5-38](#)
 - configuring automatic workload management characteristics, [5-26](#)
 - default, [5-29](#)
 - defining database roles for, [5-27](#)
 - dependencies, [5-25](#)
 - enabling event notification, [5-13](#)
 - global, [5-37](#)
 - introduction to, [1-21](#)
 - level thresholds, [5-22](#)
 - management policy
 - automatic, [5-26](#)
 - manual, [5-26](#)
 - managing after planned maintenance, [6-10](#)
 - managing groups of, [6-13](#)
 - managing on PDBs, [6-14](#)
- services (*continued*)
 - performance monitored by AWR, [5-25](#)
 - relocating, [5-19](#), [6-15](#)
 - relocating after planned maintenance, [6-10](#)
 - restricting registration with listeners, [1-19](#), [5-30](#)
 - SERVICE_NAMES parameter, [3-18](#), [5-31](#)
 - specifying a service, [5-31](#)
 - starting, [6-14](#)
 - starting after planned maintenance, [6-10](#)
 - stopping, [6-16](#)
 - stopping after planned maintenance, [6-10](#)
 - using, [5-24](#)
- services for administrator-managed databases, [3-23](#)
- session state consistency, [6-46](#)
 - auto, [6-47](#)
 - dynamic, [6-47](#)
 - static, [6-49](#)
- sessions
 - terminating on a specific instance, [3-13](#)
- setting instances, [1-33](#), [3-5](#)
- shared everything, [1-17](#)
- shared server configuration, [3-18](#)
- SHOW INSTANCE command, [3-6](#)
 - SQL*Plus, [3-6](#)
- SHOW PARAMETER command
 - SQL*Plus, [3-6](#)
- SHOW PARAMETERS command, [3-6](#)
- SHOW SGA command, [3-6](#)
 - SQL*Plus, [3-6](#)
- SHUTDOWN ABORT, [3-9](#), [B-3](#)
- SHUTDOWN command
 - ABORT option, [3-9](#)
 - SQL*Plus, [3-7](#)
- SHUTDOWN IMMEDIATE, [B-3](#)
- SHUTDOWN TRANSACTIONAL, [3-9](#)
- shutting down
 - instances, [3-9](#)
- shutting down an instance
 - abnormal shutdown, [3-9](#)
- shutting down instances, [3-9](#)
- sidalrt.log file, [B-3](#)
- single client access name
 - See SCAN
- Single Client Access Name (SCAN)
 - SRVCTL object name, [A-15](#)
- single system image, [1-19](#)
- single-instance database
 - convert to Oracle RAC database
 - administrative considerations, [15-1](#)
- single-instance databases
 - converting, [15-10](#)
 - converting to Oracle RAC database, [15-1](#)
- SMON process

- SMON process (*continued*)
- instance recovery, [8-3](#), [8-4](#)
 - recovery after SHUTDOWN ABORT, [3-9](#)
- snapshot control file, [7-2](#)
- configuring locations, [7-2](#)
- speed-up for data warehouse systems, [13-8](#)
- SPFILE
- backing up, [3-17](#)
 - backups
 - creating, [3-17](#)
 - changing parameter settings, [3-15](#)
 - corrupted, [3-16](#)
 - default names, [3-17](#)
 - location, [3-15](#)
 - naming convention for, [3-17](#)
 - recovering, [3-17](#)
 - restore with Oracle Enterprise Manager, [8-3](#)
 - restore with RMAN, [8-3](#)
 - setting values in, [3-16](#)
- SPFILE initialization parameter, [3-18](#), [3-22](#)
- SQL statements
- executing in parallel, [5-25](#)
 - instance-specific, [3-6](#)
- SQL*Plus, [3-5](#)
- commands
 - CONNECT, [3-5](#)
 - effect on instances, [3-6](#)
- SRVCTL, [1-33](#)
- administering Oracle ASM instances, [2-6](#)
 - administering services with, [5-33](#)
 - cluster database configuration tasks, [A-4](#)
 - cluster database tasks, [A-4](#)
 - command feedback, [A-1](#)
 - command syntax, [A-15](#)
 - commands
 - eval parameter, [A-1](#)
 - add database, [A-17](#)
 - add instance, [A-41](#)
 - add listener, [A-48](#)
 - add network, [A-57](#)
 - add nodeapps, [A-61](#)
 - add ons, [A-70](#)
 - add scan, [A-73](#)
 - add scan_listener, [A-79](#)
 - add service, [A-86](#)
 - add srvpool, [A-109](#)
 - add vip, [A-113](#)
 - config database, [A-20](#)
 - config listener, [A-50](#)
 - config network, [A-58](#)
 - config nodeapps, [A-62](#)
 - config ons, [A-71](#)
 - config scan, [A-74](#)
 - config scan_listener, [A-80](#)
 - config service, [A-92](#)
- SRVCTL (*continued*)
- commands (*continued*)
 - config srvpool, [A-110](#)
 - config vip, [A-114](#)
 - config volume, [A-121](#)
 - convert database, [A-21](#)
 - disable database, [A-22](#)
 - disable diskgroup, [A-35](#)
 - disable instance, [A-42](#)
 - disable listener, [A-51](#)
 - disable nodeapps, [A-63](#)
 - disable ons, [A-71](#)
 - disable scan, [A-75](#)
 - disable scan_listener, [A-81](#)
 - disable service, [A-94](#)
 - disable vip, [A-114](#)
 - disable volume, [A-122](#)
 - downgrade database, [A-23](#)
 - enable database, [A-23](#)
 - enable diskgroup, [A-36](#)
 - enable instance, [A-42](#)
 - enable listener, [A-51](#)
 - enable nodeapps, [A-63](#)
 - enable ons, [A-71](#)
 - enable scan, [A-75](#)
 - enable scan_listener, [A-81](#)
 - enable service, [A-95](#)
 - enable vip, [A-114](#)
 - enable volume, [A-123](#)
 - getenv database, [A-24](#)
 - getenv listener, [A-52](#)
 - getenv nodeapps, [A-64](#), [A-115](#)
 - help, [A-6](#)
 - modify database, [A-24](#)
 - modify instance, [A-43](#)
 - modify listener, [A-52](#)
 - modify network, [A-59](#)
 - modify nodeapps, [A-65](#)
 - modify ons, [A-71](#)
 - modify scan, [A-75](#)
 - modify scan_listener, [A-82](#)
 - modify service, [A-96](#)
 - modify srvpool, [A-111](#)
 - modify vip, [A-115](#)
 - predict database, [A-28](#)
 - predict diskgroup, [A-36](#)
 - predict listener, [A-53](#)
 - predict network, [A-60](#)
 - predict scan, [A-76](#)
 - predict scan_listener, [A-82](#)
 - predict service, [A-102](#)
 - predict vip, [A-116](#)
 - relocate database, [A-28](#)
 - relocate scan, [A-76](#)
 - relocate scan_listener, [A-83](#)

SRVCTL (*continued*)

commands (*continued*)
 relocate server, [A-85](#)
 relocate service, [A-102](#)
 relocate vip, [A-116](#)
 remove database, [A-29](#)
 remove diskgroup, [A-37](#)
 remove instance, [A-44](#)
 remove listener, [A-54](#)
 remove network, [A-60](#)
 remove nodeapps, [A-66](#)
 remove ons, [A-72](#)
 remove scan, [A-77](#)
 remove scan_listener, [A-83](#)
 remove service, [A-104](#)
 remove srvpool, [A-112](#)
 remove vip, [A-117](#)
 remove volume, [A-123](#)
 setenv database, [A-30](#), [A-118](#)
 setenv listener, [A-54](#)
 setenv nodeapps, [A-67](#)
 srvctl setenv, [3-4](#)
 start database, [3-8](#), [A-31](#)
 start diskgroup, [A-37](#)
 start home, [A-39](#)
 start instance, [3-8](#), [A-45](#)
 start listener, [A-55](#)
 start nodeapps, [A-68](#)
 start ons, [A-72](#)
 start scan, [A-78](#)
 start scan_listener, [A-84](#)
 start service, [A-105](#)
 start vip, [A-118](#)
 start volume, [A-124](#)
 status database, [A-32](#)
 status diskgroup, [A-38](#)
 status home, [A-39](#)
 status listener, [A-55](#)
 status nodeapps, [A-68](#)
 status ons, [A-73](#)
 status scan, [A-78](#)
 status scan_listener, [A-84](#)
 status server, [A-86](#)
 status service, [A-107](#)
 status srvpool, [A-112](#)
 status vip, [A-119](#)
 status volume, [A-125](#)
 stop database, [3-8](#), [A-33](#)
 stop diskgroup, [A-38](#)
 stop home, [A-40](#)
 stop instance, [A-46](#)
 stop listener, [A-56](#)
 stop nodeapps, [A-68](#)
 stop ons, [A-73](#)
 stop scan, [A-79](#)

SRVCTL (*continued*)

commands (*continued*)
 stop scan_listener, [A-85](#)
 stop service, [A-107](#)
 stop vip, [A-119](#)
 stop volume, [A-126](#)
 unsetenv database, [A-34](#),
 [A-120](#)
 unsetenv listener, [A-56](#)
 unsetenv nodeapps, [A-69](#)
 update database, [A-35](#)
 update instance, [A-47](#)
 update listener, [A-57](#)
 update scan_listener, [A-85](#)
 upgrade database, [A-35](#)
 concurrent commands, [A-1](#)
 deprecated commands and options, [A-7](#)
 deprecated commands and parameters,
 [A-15](#)
 difference between SRVCTL and
 CRSCTL, [A-7](#)
 enabling event notification, [5-13](#)
 node-level tasks, [A-4](#)
 object name
 database, [A-15](#)
 instance, [A-15](#)
 listener, [A-15](#)
 network, [A-15](#)
 node applications (nodeapps), [A-15](#)
 Oracle Notification Service, [A-15](#)
 scan, [A-15](#)
 scan_listener, [A-15](#)
 service, [A-15](#)
 vip, [A-15](#)
 object names, [A-15](#)
 overview, [3-4](#)
 overview and concepts, [1-33](#)
 single-character parameters, [A-8](#)
 specifying a continuation of command
 line entries, [A-1](#)
 stop or start cluster database, [3-8](#)
 stopping active commands, [A-7](#)
 using comma-delimited lists, [A-1](#)
 SRVM_TRACE environment variable, [B-3](#)
 Standard Edition
 installing Oracle ASM, [1-5](#)
 starting administrator-managed databases, [3-8](#)
 starting policy-managed databases, [3-8](#)
 STARTUP command
 SQL*Plus, [3-7](#)
 static session state consistency, [6-46](#)
 statistics
 contents of, [14-6](#)
 Statspack, [14-8](#)
 usage, [1-34](#)

stop database instances, [3-8](#)

storage

- administering in Oracle RAC, [2-1](#)
- cluster file system, [2-2](#)
- Oracle Automatic Storage Management (Oracle ASM), [2-5](#)

subnet

- configuring for virtual IP address, [A-4](#)

subnets, [1-19](#)

SYSASM privilege, [3-5](#)

SYSAUX tablespace

- increasing size when adding nodes, [13-7](#)
- reducing size when removing nodes, [13-7](#)

SYSDBA, [3-5](#)

SYSOPER, [3-5](#)

SYSRAC connections to an Oracle ASM instance, [3-5](#)

system change, [7-5](#)

System Global Area (SGA), [1-19](#), [14-7](#)

- size requirements, [1-19](#)

T

tablespaces

- automatic segment space management (ASSM) in Oracle RAC, [13-6](#)
- automatic undo management in Oracle RAC, [13-6](#)
- locally managed, [13-6](#)
- use in Oracle RAC, [13-6](#)

TCP network ports

- Windows Firewall considerations, [13-10](#)

TCP/IP, [1-18](#)

terminating a session on a specific instance, [3-13](#)

THREAD initialization parameter, [3-18](#)

threads

- multiple application, [5-13](#)

timed statistics, [14-6](#)

tnsnames.ora file, [3-18](#)

Top Activity drill down menu

- on the Cluster Database Performance page, [14-3](#)

Top Cluster Events, [14-7](#)

Top Cluster Events, ASH report, [14-7](#)

Top Remote Instance, [14-7](#)

Top Remote Instance, ASH report, [14-7](#)

trace files, [B-1](#)

- managing, [B-1](#)
- sidaIrt.log, [B-3](#)

trace files for background processes, [B-1](#)

TRACE_ENABLED initialization parameter, [3-22](#)

tracing

- enabling Java-based tools and utilities, [B-3](#)
- SRVM_TRACE environment variable, [B-3](#)
- writing to log files, [B-3](#)

transaction failover, [6-53](#)

Transaction Guard, [6-53](#)

- configuring JDBC-thin clients, [5-11](#), [5-12](#)
- configuring OCI clients, [5-14](#)
- configuring service attributes, [5-34](#)
- configuring services for, [6-54](#)
- introduction to, [1-21](#)
- transaction history table, [6-54](#)

transaction history table, [6-54](#)

transaction idempotence, [6-53](#)

transactional TAF, [5-14](#)

transactions

- distributed SQL, [5-16](#)
- DTP/XA, [5-16](#)
- instance failure, [8-3](#)
- rolling back, [3-13](#), [8-3](#)
- waiting for recovery, [8-3](#)

Transparent Application Continuity, [6-23](#)

transparent application failover (TAF)

- and services, [6-55](#)

Transparent Data Encryption

- specifying the
 - ENCRYPTION_WALLET_LOCATION parameter, [13-8](#)
- with encrypted keystores and obfuscated keystores, [13-8](#)

tuning

- overview, [1-35](#)
- using ADDM, [1-34](#)

U

undo tablespace, [3-1](#)

UNDO_MANAGEMENT initialization parameter, [3-21](#)

UNDO_RETENTION initialization parameter, [3-22](#)

UNDO_TABLESPACE parameter, [3-21](#)

upgrades

- changing management policy for, [3-28](#)

User Datagram Protocol (UDP), [B-4](#)

user process trace files, [B-1](#)

V

V\$, [1-34](#)

V\$ view, [14-5](#)

V\$CLUSTER_INTERCONNECTS, [14-4](#), [B-4](#)

V\$CONFIGURED_INTERCONNECTS, [14-4](#)

valid node checking, [1-19](#), [5-30](#)

vendor clusterware, [1-1](#)

verification

- data files, online files, [2-2](#)

versions

versions (*continued*)
compatibility for Oracle RAC and Oracle Database software, [1-4](#)

views
creating for Oracle Real Application Clusters, [14-6](#)
dynamic performance
for performance monitoring, [14-5](#)
GV\$, [14-5](#)
for performance monitoring, [14-5](#)
GV\$SESSION, [3-13](#)
instance-specific, [14-5](#)
V\$ views, [14-5](#)

VIP
SRVCTL object name, [A-15](#)

VIPs
node, [1-18](#)

Virtual Internet Protocol (VIP) address, [1-1](#)

virtual IP address
requirements, [A-4](#)

VNCR
See valid node checking

W

wait events, [14-8](#)

wait events (*continued*)
aggregated for service performance, [14-3](#)
block-related, [14-11](#)
contention-related, [14-12](#)
load-related, [14-12](#)
message-related, [14-11](#)

Welcome page, [11-4](#)

Windows Firewall, [13-10](#)

workload management
See automatic workload management

workloads
and services, [5-24](#)
See *Also* automatic workload management, [5-24](#)

X

XA affinity, [5-17](#)

XA transactions, [5-16](#)
spanning Oracle RAC instances, [5-16](#)
using services with, [5-18](#)

XA Transactions, [13-7](#)