

Oracle® Data Guard Broker



21c
F31839-04
April 2022

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

ORACLE®

Oracle Data Guard Broker, 21c

F31839-04

Copyright © 1999, 2020, Oracle and/or its affiliates.

Primary Author: Padmaja Potineni

Contributors: Kathy Rich, Larry Carpenter, Laurence Clarke, Jeff Detjen, Mahesh Girkar, Nitin Karkhanis, Youngtae Kim, Sadhana Kyathappala, Chang Kyu Lee, Steve Lee, Jiangbin Luo, Bob McGuirk, Joe Meeks, Darryl Presley, Ashish Ray, Pieter Van Puymbroeck, Lawrence To, Nick Wagner, Zaixian Xie, Di Yang

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 embedded, installed or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software" or "commercial computer software documentation" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. 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, Java, and MySQL are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside 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, Epyc, and the AMD 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	xv
Documentation Accessibility	xv
Related Documents	xv
Conventions	xv

1 Oracle Data Guard Broker Concepts

Changes in Oracle Database Release 21c for Oracle Data Guard Broker	1-1
Overview of Oracle Data Guard and the Broker	1-2
Oracle Data Guard Configurations and Broker Configurations	1-3
Oracle Data Guard Broker	1-3
Oracle Data Guard Broker in a Multitenant Environment	1-5
Oracle Data Guard Broker and Oracle Global Data Services	1-5
Benefits of Oracle Data Guard Broker	1-6
Oracle Data Guard Broker Components	1-9
Oracle Data Guard Broker User Interfaces	1-10
Oracle Enterprise Manager Cloud Control	1-10
Oracle Data Guard Command-Line Interface (DGMGRL)	1-11
Oracle Data Guard Monitor	1-11
Oracle Data Guard Monitor (DMON) Process	1-12
Configuration Management	1-14
Database Property Management	1-14

2 Oracle Data Guard Installation

Oracle Data Guard Installation	2-1
Prerequisites	2-2

3 Managing Broker Configurations

Overview of Broker Configurations	3-1
Configuration Support	3-1

Configuration Support for DG PDB	3-4
DG PDB Terminology	3-4
Components of DG PDB Configuration	3-4
About the DGPDB_INT User Account	3-6
DG PDB Configuration Restrictions	3-6
Workflow for Using DG PDB Configuration	3-6
Configuration Properties	3-7
Setting Up the Broker Configuration Files	3-7
Renaming the Broker Configuration Files	3-8
Managing Broker Configuration Files in an Oracle RAC Environment	3-9
Using Cluster File System (CFS) for Configuration Files	3-9
Using Oracle ASM Disk Groups for Configuration Files	3-9
Starting the Data Guard Broker	3-10
Management Cycle of a Broker Configuration	3-11
Enable and Disable Operations	3-14
Configuration Status	3-15

4 Managing the Members of a Broker Configuration

Managing Broker Configuration Members	4-1
Managing States of Broker Configuration Members	4-1
Database State Transitions	4-3
Managing Database Properties	4-5
Monitorable (Read-Only) Properties	4-7
Configurable (Changeable) Properties	4-7
Resetting Broker Configurable Properties to Default Values	4-8
Managing Redo Transport Services	4-8
Setting Up For Redo Transport	4-9
Managing Redo Transport Services for Data Protection Modes	4-9
Advanced Redo Transport Settings	4-10
Turning Redo Transport Services On and Off	4-13
Specifying Locations for Archived Redo Log Files	4-14
Other Redo Transport Settings	4-15
Redo Transport Services in an Oracle RAC Database Environment	4-16
Transport Lag	4-16
Managing Redo Transport Services for Recovery Appliance	4-17
Managing Log Apply Services	4-19
Managing Delayed Apply	4-19
Managing Parallel Apply with Redo Apply	4-20
Managing Multi-Instance Redo Apply	4-20
Apply Services in an Oracle RAC Database Environment	4-21

Selecting the Apply Instance	4-21
Apply Instance Failover	4-23
Apply Lag	4-23
Managing Data Protection Modes	4-24
Setting the Protection Mode for Your Configuration	4-24
Setting the Protection Mode Task 1: Determine Which Data Protection Mode You Want to Use	4-25
Setting the Protection Mode Task 2: Set up standby redo log files	4-26
Setting the Protection Mode Task 3: Set the redo transport mode	4-27
Setting the Protection Mode Task 4: Using DGMGRL or Cloud Control	4-27
How the Protection Modes Influence Broker Operations	4-28
Upgrading or Downgrading the Current Protection Mode	4-28
Switchover Operations	4-29
Failover Operations	4-29
Disable and Enable Operations	4-30
Requirements For Removing a Database from the Configuration	4-30
Requirements On Other Operations	4-31
Managing Far Sync Instances	4-31
Managing Fast-Start Failover	4-33
Configure Properties to Tune Fast-Start Failover	4-33
Configure Conditions for Fast-start Failover	4-35
Application Initiated Fast-Start Failover	4-36
Managing Database Conversions	4-36
Database Status	4-37
Querying Database Status	4-37
Validating a Database Before a Role Change	4-39
Validating the Server Parameter Files Before a Role Change	4-39
Validating the Network Configuration Before a Role Change	4-40
Validating the Static Connect Identifier Before a Role Change	4-40
Validating the DGConnectIdentifier Property	4-40

5 Managing Members of a DG PDB Broker Configuration

Managing Broker Configuration Members	5-1
Managing States of Broker Configuration Members	5-2
State Transitions for DG PDBs	5-3
Managing Database Properties in DG PDB Configurations	5-3
Managing Redo Transport Services for DG PDBs	5-4
Viewing Transport Lag and Apply Lag	5-4
Monitoring a DG PDB Configuration	5-5

6 Switchover and Failover Operations

Overview of Switchover and Failover in a Broker Environment	6-1
Choosing a Target Standby Database	6-2
Choosing a Target Standby Database for Switchover	6-3
Choosing a Target Standby Database for Failover	6-3
Switchover	6-4
Before You Perform a Switchover Operation	6-5
Starting a Switchover	6-6
How the Broker Performs a Switchover	6-7
Manual Failover	6-8
Complete and Immediate Manual Failovers	6-8
Performing a Manual Failover Operation	6-10
Performing a Manual Failover Task 1: Determine Which of the Available Standby Databases is the Best Target for the Failover	6-10
Performing a Manual Failover Task 2: Start the Failover	6-10
Performing a Manual Failover Task 3: Reset the Protection Mode	6-11
Performing a Manual Failover Task 4: Re-establish a Disaster-Recovery Configuration	6-12
How the Broker Performs a Complete Failover Operation	6-12
How the Broker Performs an Immediate Failover Operation	6-14
Reenabling Disabled Databases After a Role Change	6-15
How to Reinststate a Database	6-15
How to Re-create and Reenable a Disabled Database	6-17
Fast-Start Failover	6-17
About Fast-start Failover	6-17
Fast-start Failover Protection Modes	6-18
Observers in a Fast-start Failover Configuration	6-19
Location of Client-side Broker Files	6-21
Fast-start Failover Callout Configuration Files	6-23
Prerequisites for Enabling Fast-Start Failover	6-23
Enabling Fast-Start Failover	6-24
Enabling Fast-Start Failover Task 1: Determine Which Available Standby Databases Should Be Targets for the Failover	6-25
Enabling Fast-Start Failover Task 2: Specify Target Standby Databases with the FastStartFailoverTarget Configuration Property	6-25
Enabling Fast-Start Failover Task 3: Determine the Protection Mode You Want	6-25
Enabling Fast-Start Failover Task 4: Set the FastStartFailoverThreshold Configuration Property	6-26
Enabling Fast-Start Failover Task 5: Set Other Properties Related to Fast-Start Failover (Optional)	6-27

Enabling Fast-Start Failover Task 6: Enable Additional Fast-Start Failover Conditions (Optional)	6-29
Enabling Fast-Start Failover Task 7: Configure Actions Before and After Fast-start Failover (Optional)	6-30
Enabling Fast-Start Failover Task 8: Using DGMGRL or Cloud Control	6-31
Enabling Fast-Start Failover Task 9: Start the Observer	6-32
Enabling Fast-Start Failover Task 10: Verify the Fast-Start Failover Environment When Fast-Start Failover Is Enabled and the Observer Is Running	6-34
Restrictions When Fast-Start Failover is Enabled	6-36
Configuring Fast-Start Failover in Observe-only Mode	6-37
Shutting Down the Primary Database When Fast-Start Failover Is Enabled	6-38
Performing Manual Role Changes When Fast-Start Failover Is Enabled	6-38
Directing a Fast-Start Failover From an Application	6-38
Viewing Fast-Start Failover Configuration Statistics and Status	6-39
V\$DATABASE View	6-41
V\$FS_FAILOVER_STATS View	6-44
Disabling Fast-Start Failover	6-45
Performance Considerations for Fast-Start Failover	6-48
Managing the Observer	6-48
Installing and Starting the Observer	6-49
Viewing Information About the Master Observer	6-52
Viewing Information About All Observers	6-53
What Happens if the Master Observer Fails?	6-53
Managing Observer's Connection to the Primary	6-54
Configuring the Time Taken to Initiate Fast-Start Failover	6-55
Stopping the Observer	6-55
Moving the Observer to Another Computer	6-57
How the Observer Maintains Fast-Start Failover Configuration Information	6-57
Managing Observers for Multiple Configurations	6-57
Patching an Environment When the Observer Is Running and Fast-start Failover Is Enabled	6-61
Reinstating the Former Primary Database in the Broker Configuration	6-62
Requirements	6-62
Restrictions on Reinstatement	6-62
How the Broker Handles a Failed Reinstatement	6-63
Shutting Down Databases In a Fast-Start Failover Environment	6-63
Database Client Considerations	6-63
Oracle Data Guard Specific FAN and FCF Configuration Requirements	6-64
Oracle Net Configuration Requirements	6-65
Database Service Configuration Requirements	6-65
ONS Configuration Requirements	6-68

7 Switchover and Failover in DG PDB Environments

Overview of Switchover and Failover in a DG PDB Environment	7-1
Performing PDB Switchover	7-1
How Broker Performs Switchover in a DG PDB Environment	7-1
Starting a Switchover to a PDB	7-2
Performing PDB Failover	7-2
Before You Begin DG PDB Failover	7-3
How Broker Performs Failover in a DG PDB Environment	7-3
Starting a PDB Failover	7-3

8 Scenarios Using the DGMGRL Command-Line Interface

Prerequisites for Getting Started	8-1
Scenario 1: Creating a Configuration	8-2
Creating a Configuration Task 1: Invoke DGMGRL	8-2
Creating a Configuration Task 2: Connect to the Primary Database	8-3
Creating a Configuration Task 3: Clear Existing Remote Redo Transport Destinations on Standbys and Far Sync Instances To Be Added.	8-3
Creating a Configuration Task 4: Create the Broker Configuration	8-4
Creating a Configuration Task 5: Show the Configuration Information	8-4
Creating a Configuration Task 6: Add a Standby Database to the Configuration	8-5
Scenario 2: Setting Database Properties	8-5
Scenario 3: Enabling the Configuration and Databases	8-7
Scenario 4: Setting the Configuration Protection Mode	8-8
Scenario 5: Setting up Maximum Availability Mode with a Far Sync Instance	8-10
Scenario 6: Enabling Fast-Start Failover and Starting the Observer	8-12
Scenario 7: Enabling Fast-Start Failover When a Far Sync Instance Is In Use	8-15
Scenario 8: Performing Routine Management Tasks	8-15
Changing Properties and States	8-15
Alter a Database Property	8-15
Reset a Property to Its Default Value	8-16
Alter the State of a Standby Database	8-16
Alter the State of a Primary Database	8-16
Disabling the Configuration and Databases	8-16
Disable a Configuration	8-17
Disable a Standby Database	8-17
Disabling a Far Sync Instance	8-18
Removing the Configuration, a Standby Database, or a Far Sync Instance	8-19
Removing a Standby Database from the Configuration	8-19

Removing a Far Sync Instance from the Configuration	8-20
Removing a Broker Configuration	8-20
Scenario 9: Performing a Switchover Operation	8-21
Using the SWITCHOVER Command Task 1: Check the Primary Database	8-22
Using the SWITCHOVER Command Task 2: Check the Standby Database That is the Target of the Switchover	8-23
Using the SWITCHOVER Command Task 3: Confirm That the Database Is Ready for a Role Change	8-23
Using the SWITCHOVER Command Task 4: Issue the Switchover Command	8-25
Using the SWITCHOVER Command Task 5: Show the Configuration	8-26
Scenario 10: Performing a Manual Failover Operation	8-26
Scenario 11: Reinstating a Failed Primary Database	8-28
Scenario 12: Converting a Physical Standby to a Snapshot Standby	8-30
Scenario 13: Monitoring a Data Guard Configuration	8-31
Monitoring a Configuration Task 1: Check the Configuration Status	8-31
Monitoring a Configuration Task 2: Check the Database Status	8-32
Monitoring a Configuration Task 3: Check the LogXptStatus Monitorable Property	8-32
Monitoring a Configuration Task 4: Check the InconsistentLogXptProps Monitorable Property	8-32
Scenario 14: Adding a Recovery Appliance to a Broker Configuration	8-33
Scenario 15: Exporting and Importing a Broker Configuration File	8-33
Exporting a Broker Configuration	8-34
Importing a Broker Configuration	8-34
Scenario 16: Using the Observe-only Mode for Fast-Start Failover	8-35
Configuring Observe-only Mode for Fast-Start Failover	8-35
Sample Content of the Log Files in Observe-only Mode	8-36
Disabling Observe-only Mode for Fast-start Failover	8-36

9 Scenarios for Using DGMGRL with a DG PDB Configuration

Prerequisites for Using DG PDB	9-1
Scenario 1: Create the Source and Target Configuration	9-2
Scenario 2: Establish a Connection Between the Configurations	9-2
Scenario 3: Setting Up Data Guard for the Source PDB	9-2
Scenario 4: Enable the DG PDB Configuration and Start Redo Transport	9-3
Scenario 5: Switchover Source PDB to Target PDB	9-3
Scenario 6: Failover to Target PDB	9-5
Scenario 7: Monitoring a DG PDB Configuration	9-7

10 Oracle Data Guard Command-Line Interface Reference

Starting the Data Guard Command-Line Interface	10-1
DGMGRL Optional Parameters	10-1
DGMGRL Command Format and Parameters	10-3
DGMGRL Command Usage Notes	10-7
Exiting the Data Guard Command-Line Interface	10-9
@ (at sign) Command	10-9
/ (slash) Command	10-10
ADD CONFIGURATION	10-11
ADD DATABASE	10-12
ADD FAR_SYNC	10-13
ADD PLUGGABLE DATABASE	10-13
ADD RECOVERY_APPLIANCE	10-15
CONNECT	10-16
CONVERT DATABASE	10-17
CREATE CONFIGURATION	10-19
CREATE FAR_SYNC	10-20
DISABLE CONFIGURATION	10-22
DISABLE DATABASE	10-23
DISABLE FAR_SYNC	10-23
DISABLE FAST_START FAILOVER	10-24
DISABLE FAST_START FAILOVER CONDITION	10-25
DISABLE RECOVERY_APPLIANCE	10-25
EDIT CONFIGURATION (Property)	10-26
EDIT CONFIGURATION (Protection Mode)	10-26
EDIT CONFIGURATION (RENAME)	10-28
EDIT CONFIGURATION RESET (Property)	10-29
EDIT CONFIGURATION (Set Source)	10-29
EDIT DATABASE (Property)	10-30
EDIT DATABASE (Parameter)	10-32
EDIT DATABASE (Rename)	10-33
EDIT DATABASE (State)	10-34
EDIT DATABASE RESET (Property)	10-34
EDIT DATABASE RESET (Parameter)	10-35
EDIT FAR_SYNC	10-35
EDIT FAR_SYNC RESET (Property)	10-37
EDIT FAR_SYNC RESET (Parameter)	10-37
EDIT PLUGGABLE DATABASE (State)	10-38
EDIT RECOVERY_APPLIANCE (Property)	10-38
EDIT RECOVERY_APPLIANCE (Parameter)	10-39

EDIT RECOVERY_APPLIANCE (Rename)	10-40
EDIT RECOVERY_APPLIANCE RESET (Property)	10-41
EDIT RECOVERY_APPLIANCE RESET (Parameter)	10-41
ENABLE CONFIGURATION	10-42
ENABLE DATABASE	10-43
ENABLE FAR_SYNC	10-44
ENABLE FAST_START FAILOVER	10-44
ENABLE FAST_START FAILOVER CONDITION	10-46
ENABLE RECOVERY_APPLIANCE	10-48
EXIT	10-48
EXPORT CONFIGURATION	10-49
FAILOVER	10-49
FAILOVER TO PLUGGABLE DATABASE	10-52
HELP	10-53
HOST or ! (exclamation point)	10-54
IMPORT CONFIGURATION	10-55
MIGRATE PLUGGABLE DATABASE	10-56
PREPARE DATABASE FOR DATA GUARD	10-60
QUIT	10-63
REINSTATE DATABASE	10-63
REMOVE CONFIGURATION	10-64
REMOVE DATABASE	10-65
REMOVE FAR_SYNC	10-66
REMOVE INSTANCE	10-67
REMOVE PLUGGABLE DATABASE	10-67
REMOVE RECOVERY_APPLIANCE	10-68
SET ECHO	10-69
SET FAST_START FAILOVER TARGET	10-70
SET MASTEROBSERVER TO	10-72
SET MASTEROBSERVERHOSTS	10-73
SET ObserverConfigFile	10-74
SET TIME	10-74
SET TRACE_LEVEL	10-75
SHOW ALL	10-76
SHOW CONFIGURATION	10-76
SHOW CONFIGURATION WHEN PRIMARY IS	10-79
SHOW DATABASE	10-81
SHOW FAR_SYNC	10-85
SHOW FAST_START FAILOVER	10-86
SHOW INSTANCE	10-88
SHOW OBSERVER	10-90

SHOW ObserverConfigFile	10-91
SHOW OBSERVERS	10-91
SHOW PLUGGABLE DATABASE	10-92
SHOW RECOVERY_APPLIANCE	10-93
SHUTDOWN	10-95
SPOOL	10-96
SQL	10-97
START OBSERVER	10-97
START OBSERVER IN BACKGROUND	10-100
START OBSERVING	10-102
STARTUP	10-103
STOP OBSERVER	10-105
STOP OBSERVING	10-106
SWITCHOVER	10-107
SWITCHOVER PLUGGABLE DATABASE	10-112
VALIDATE DATABASE	10-113
VALIDATE DATABASE DATAFILE	10-117
VALIDATE DATABASE SPFILE	10-118
VALIDATE DGConnectIdentifier	10-120
VALIDATE FAR_SYNC	10-122
VALIDATE FAST_START FAILOVER	10-124
VALIDATE NETWORK CONFIGURATION	10-125
VALIDATE STATIC CONNECT IDENTIFIER	10-127

11 Oracle Data Guard Broker Properties

Configuration Properties	11-1
BystandersFollowRoleChange	11-3
CommunicationTimeout	11-4
ConfigurationSimpleName	11-4
ConfigurationWideServiceName	11-5
ExternalDestination1	11-6
ExternalDestination2	11-7
FastStartFailoverAutoReinstate	11-8
FastStartFailoverLagLimit	11-8
FastStartFailoverPmyShutdown	11-9
FastStartFailoverThreshold	11-9
ObserverOverride	11-10
ObserverPingInterval	11-11
ObserverPingRetry	11-11
ObserverReconnect	11-12

OperationTimeout	11-12
PreferredObserverHosts	11-13
PrimaryLostWriteAction	11-14
TraceLevel	11-15
Monitorable (Read-Only) Properties	11-16
InconsistentLogXptProps (Inconsistent Redo Transport Properties)	11-16
InconsistentProperties (Inconsistent Properties)	11-17
LogXptStatus (Redo Transport Status)	11-18
LsbyFailedTxnInfo (Logical Standby Failed Transaction Information)	11-18
LsbyParameters (Logical Standby Parameters)	11-19
RecvQEntries (Receive Queue Entries)	11-19
SendQEntries (Send Queue Entries)	11-20
TopWaitEvents	11-22
Configurable Properties	11-22
AlternateLocation	11-24
ArchiveLocation	11-25
ApplyInstances	11-26
ApplyInstanceTimeout	11-27
ApplyLagThreshold	11-27
ApplyParallel	11-28
Binding	11-28
DelayMins	11-29
DGConnectIdentifier	11-30
Encryption	11-31
FastStartFailoverTarget	11-32
InstanceName	11-33
LogShipping	11-33
LogXptMode	11-34
MaxFailure	11-35
NetTimeout	11-36
ObserverConnectIdentifier	11-37
PreferredApplyInstance	11-37
PreferredObserverHosts	11-38
RedoCompression	11-39
RedoRoutes	11-39
Redo Routing Rules	11-40
ReopenSecs	11-43
StandbyAlternateLocation	11-43
StandbyArchiveLocation	11-44
StaticConnectIdentifier	11-45
TransportDisconnectedThreshold	11-46

12 Troubleshooting Oracle Data Guard

Sources of Diagnostic Information	12-1
General Problems and Solutions	12-1
ORA-16596: database not part of the Oracle Data Guard broker configuration	12-2
Redo Accumulating on the Primary Is Not Sent to Some Standby Databases	12-2
Many Log Files Are Received on a Standby Database But Not Applied	12-3
The Request Timed Out or Cloud Control Performance Is Sluggish	12-4
The Primary Database is Flashed Back	12-4
Standby Fails to Automatically Start Up Due to Unknown Service (ORA-12514)	12-4
Troubleshooting Problems During a Switchover Operation	12-5
Troubleshooting Problems During a Failover Operation	12-5
Failed Failovers to Physical Standby Databases	12-5
Failed Broker Complete Physical Failovers	12-6
Failed Broker Immediate Physical Failovers	12-6
Failed Failovers to Logical Standby Databases	12-6
Troubleshooting Problems with the Observer	12-7
Problems Because the Observer Has Stopped	12-7
Capturing Observer Actions in the Observer Log File	12-8

A Oracle Data Guard Broker Upgrading and Downgrading

Considerations While Using the DBMS_ROLLING Package	A-1
---	-----

Glossary

Index

Preface

This document provides information about Oracle Data Guard broker, a management and monitoring interface that helps you configure, monitor, and control an Oracle Data Guard broker configuration.

Audience

Oracle Data Guard Broker is intended for database administrators (DBAs) and system administrators who want to use the Oracle Data Guard broker to automate many of the tasks involved in configuring and monitoring an Oracle Data Guard configuration.

The information presented in this book assumes that you are already familiar with Oracle Data Guard, Oracle Enterprise Manager Cloud Control (Cloud Control), Oracle Real Application Clusters (Oracle RAC), and the network services provided by Oracle Net Services.

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

Refer to the following documentation for more information about Oracle Data Guard:

- *Oracle Data Guard Concepts and Administration*.
- Oracle release notes specific to your operating system.
- Oracle installation guide specific to your operating system.
- For more information about managing Oracle Data Guard through the Cloud Control graphical user interface (GUI), see the Cloud Control online Help.

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.

1

Oracle Data Guard Broker Concepts

Learn about Oracle Data Guard broker, its architecture and components, and how it automates the creation, control, and monitoring of an Oracle Data Guard configuration.

- [Overview of Oracle Data Guard and the Broker](#)
- [Benefits of Oracle Data Guard Broker](#)
- [Oracle Data Guard Broker Components](#)
- [Oracle Data Guard Broker User Interfaces](#)
- [Oracle Data Guard Monitor](#)

See *Oracle Data Guard Concepts and Administration* for the definition of an Oracle Data Guard configuration and for complete information about Oracle Data Guard concepts and terminology.



Note:

A multitenant container database is the only supported architecture in Oracle Database 21c. While the documentation is being revised, legacy terminology may persist. In most cases, "database" and "non-CDB" refer to a CDB or PDB, depending on context. In some contexts, such as upgrades, "non-CDB" refers to a non-CDB from a previous release.

Changes in Oracle Database Release 21c for Oracle Data Guard Broker

These are the changes in *Oracle Data Guard Broker* for Oracle Database Release 21c.

New Features

- Oracle Data Guard Broker supports Data Guard for Pluggable Databases.
- Misconfigurations in a fast-start failover configuration can be identified by validating the configuration using the `VALIDATE FAST_START FAILOVER` command. See [VALIDATE FAST_START FAILOVER](#).
- A default directory, managed by Data Guard broker, is used to store the observer runtime data file, observer files, and callout configuration files. See [Location of Client-side Broker Files](#).
- Callout configuration scripts can be used to automatically execute specified tasks before and after a fast-start failover operation. See [Fast-start Failover Callout Configuration Files](#).
- When no synchronous standby destinations are available, a standby that uses asynchronous redo transport can be used as a fast-start failover target provided the

`FastStartFailoverLagLimit` configuration property is set. When a synchronous standby becomes available, the broker automatically switches back to the synchronous standby.

- Fast-start failover can be enabled in maximum availability mode for configurations with a `FAR SYNC` instance, even if the primary database has no synchronous standby databases.
- The syntax for the `START OBSERVER`, `START OBSERVER IN BACKGROUND`, and `STARTUP` commands contain changes that allow more flexibility in specifying command options. See [START OBSERVER](#), [START OBSERVER IN BACKGROUND](#), and [STARTUP](#).
- The `PreferredObserverHosts` database configurable property has changed to allow the user to specify which observer should observe the configuration when a database is in the primary role. The user can also assign priorities to the observer.
- The maximum number of observers in a Data Guard Broker configuration is increased to four.
- The `CREATE FAR_SYNC` command instantiates a far sync instance and adds it to the broker configuration. See [CREATE FAR_SYNC](#).
- The `PREPARE DATABASE FOR DATA GUARD` command configures a database and sets it up to be used as a primary database in a Data Guard broker configuration. See [PREPARE DATABASE FOR DATA GUARD](#).
- Logical standby databases

New data types added after Oracle Database 12c Release 1 (12.1) are not supported with Oracle Data Guard logical standby. For example, Oracle Data Guard logical standby does not support long identifiers, complex Abstract Data Types (ADTs), and spatial data types. Note that this limitation does not exist with an Oracle Data Guard physical standby database, `DBMS_ROLLING`, or Oracle GoldenGate. To obtain the benefits of a standby database with more recent data types, Oracle recommends that you consider using either a physical standby database, a snapshot standby database, or that you use the logical replication features of Oracle GoldenGate.

Desupported Features

- The following parameters that were deprecated in Oracle Database Release 19c are desupported: `ArchiveLagTarget`, `DataGuardSyncLatency`, `LogArchiveMaxProcesses`, `LogArchiveMinSucceedDest`, `LogArchiveTrace`, `StandbyFileManagement`, `DbFileNameConvert`, `LogArchiveFormat`, `LogFileNameConvert`, `LsbyMaxEventsRecorded`, `LsbyMaxServers`, `LsbyMaxSga`, `LsbyPreserveCommitOrder`, `LsbyRecordAppliedDdl`, `LsbyRecordSkipDdl`, `LsbyRecordSkipErrors`, and `LsbyParameters`.

Overview of Oracle Data Guard and the Broker

Oracle Data Guard ensures high availability, data protection, and disaster recovery for enterprise data. Oracle Data Guard provides a comprehensive set of services that create, maintain, manage, and monitor one or more standby databases to enable production Oracle databases to survive disasters and data corruptions. Oracle Data Guard maintains these standby databases as transactionally consistent copies of the primary database. If the primary database becomes unavailable because of a planned or an unplanned outage, Oracle Data Guard enables you to switch any standby

database to the production role, thus minimizing the downtime associated with the outage. Oracle Data Guard can be used with traditional backup, recovery, and cluster techniques, as well as with the Flashback Database feature to provide a high level of data protection and data availability.

Oracle Data Guard Configurations and Broker Configurations

An *Oracle Data Guard configuration* consists of one primary database and a combination of standby databases, far sync instances, and Zero Data Loss Recovery Appliances that receive redo directly from the primary database. The members of an Oracle Data Guard configuration are connected by Oracle Net and may be dispersed geographically. There are no restrictions on where the members are located as long as they can communicate with each other.

The Oracle Data Guard broker logically groups these members into a *broker configuration* that allows the broker to manage and monitor them together as an integrated unit. You can manage a broker configuration using either Oracle Enterprise Manager Cloud Control (Cloud Control) or the Oracle Data Guard command-line interface.

Oracle Data Guard Broker

The Oracle Data Guard broker is a distributed management framework that automates and centralizes the creation, maintenance, and monitoring of Oracle Data Guard configurations.

The following list describes some of the operations the broker automates and simplifies:

- Creating Oracle Data Guard configurations that include one primary database and zero or more standbys (physical, logical, or snapshot), far sync instances, and Zero Data Loss Recovery Appliances along with any associated redo transport services and log apply services. Databases can be single-instance, Oracle Real Application Clusters (Oracle RAC), or Oracle RAC One Node databases.
- Adding standbys (physical, logical, or snapshot), far sync instances, and Zero Data Loss Recovery Appliances to an existing Data Guard configuration.
- Managing the protection mode for the configuration.
- Invoking switchover or failover with a single command to initiate and control complex role changes across all databases in the configuration.
- Configuring failover to occur automatically upon loss of the primary database, increasing availability without manual intervention.
- Monitoring the status of the entire configuration, capturing diagnostic information, reporting statistics such as the Redo Apply rate and the redo generation rate, and detecting problems quickly with centralized monitoring, testing, and performance tools.
- Assessing whether a database is ready to become a primary. (See "[VALIDATE DATABASE](#)").
- Assessing whether the network is properly configured between databases. (See "[VALIDATE NETWORK CONFIGURATION](#)".)

You can perform all management operations locally or remotely through the broker's easy-to-use interfaces: the Oracle Data Guard management pages in Cloud Control, and the Oracle Data Guard command-line interface called DGMGRL.

These interfaces simplify the configuration and management of an Oracle Data Guard configuration. The following table provides a comparison of configuration management using the broker's interfaces and using SQL*Plus.

Table 1-1 Configuration Management With and Without the Broker

Tasks	With the Broker	Without the Broker
General	Provides primary database, standby database, and far sync instance management as one unified configuration.	You must manage the primary database, standby databases, and far sync instances separately.
Standby Database Creation	Provides Cloud Control wizards that automate and simplify the steps required to create a configuration with an Oracle database on each site, including creating the standby control file, online redo log files, datafiles, and server parameter files.	You must manually (using RMAN or other tools, such as DBCA): <ul style="list-style-type: none"> • Create the standby. • Copy the database files to the standby database system. • Create a control file on the standby database system. • Create server parameter or initialization parameter files on the standby database system.
Configuration and Management	Enables you to configure and manage multiple databases from a single location and automatically unifies all of the databases in the broker configuration.	You must manually connect to multiple locations in order to: <ul style="list-style-type: none"> • Set up redo transport services and log apply services on each database in the configuration. • Manage the primary database and standby databases individually.
Control	<ul style="list-style-type: none"> • Automatically sets up redo transport services and log apply services. Simplifies management of these services, especially in an Oracle RAC environment or for configurations that include large numbers of standby databases. • Simplifies switchovers, failovers, reinstatements, and conversions to and from a snapshot standby database, allowing you to invoke them through a single command. • Automates failover by allowing the broker to determine if failover is necessary and to initiate failover to a specified target standby database, with no need for DBA intervention and with either no loss of data or with a configurable amount of data loss. • Integrates role changes with Oracle Clusterware. • Manageable and monitorable through Cloud Control • Integrates with Oracle Global Data Services, providing support for role-specific global services. 	You must manually: <ul style="list-style-type: none"> • Use multiple SQL statements to manage the database. • Coordinate sequences of multiple commands across multiple database sites to execute switchover and failover operations. • Coordinate sequences of multiple commands to manage services and instances during role transitions.
Monitoring	<ul style="list-style-type: none"> • Provides continuous monitoring of the configuration health, database health, and other runtime parameters. • Provides a unified updated status and detailed reports. • Provides integration with Cloud Control events. 	You must manually: <ul style="list-style-type: none"> • Monitor the status and runtime parameters using fixed views on each database—there is no unified view of status for all of the databases in the configuration. • Provide a custom method for monitoring Cloud Control events.

Oracle Data Guard Broker in a Multitenant Environment

Oracle Data Guard broker supports multitenant container databases (CDBs) within a broker configuration.

Keep the following in mind:

- When the primary database is a CDB, all standby databases in the broker configuration must also be CDBs.
- When a broker configuration is made up of CDBs, all broker actions execute at the root level and not at the individual pluggable database (PDB) level.
- To administer a multitenant environment you must have the `CDB_DBA` role.
- After a role change, the broker opens all active instances of the new primary. The broker also opens all active instances of a snapshot standby, a logical standby and the first instance of a physical standby (if the former physical standby was open for read).
- The broker also opens all PDBs on all instances it opens.

See Also:

- *Oracle Database Concepts* for more information about CDBs

Oracle Data Guard Broker and Oracle Global Data Services

To manage broker configurations that support Oracle Global Data Services (GDS), you use both the broker command-line interface, DGMGRL, and the GDS command line interface, GDSCTL. Broker configurations are added to GDS pools (logical groupings of GDS databases which span across GDS regions).

Managing Broker Configurations in a GDS Pool

To add a broker configuration to a GDS pool, use the GDS command-line interface, GDSCTL. A broker configuration can only be added to an empty pool. A broker configuration cannot span multiple pools.

To delete a broker configuration that is in a pool, you must first remove it from the pool by using the `GDSCTL REMOVE BROKERCONFIG ...` command. Otherwise an error is returned.

Only entire broker configurations are added and removed using GDSCTL; to add and remove individual databases to/from a broker configuration, use the broker command-line interface, DGMGRL.

Managing Individual Configuration Databases When They Are Part of a GDS Pool

If a GDS pool already contains a broker configuration, then only databases that belong to that configuration can be added to the pool. The only way to add or remove individual databases to or from a broker configuration in a pool is to use the broker command-line interface, DGMGRL.

When a database is added to a broker configuration, that database is automatically also added to the GDS pool to which the configuration belongs.

When a database is removed from a broker configuration, it is automatically removed from the GDS pool to which the configuration belongs.

The broker interacts with GDS and notifies it of any role or configuration changes to ensure that the appropriate database services are active and that the appropriate FAN events are published after a role change.

 **See Also:**

- *Oracle Database Global Data Services Concepts and Administration Guide* for more information about GDS.

Benefits of Oracle Data Guard Broker

The broker's interfaces improve usability and centralize management and monitoring of an Oracle Data Guard configuration.

Available as a feature of the Enterprise Edition and Personal Edition of the Oracle database, the broker is also integrated with the Oracle database and Cloud Control. These broker attributes result in the following benefits:

Disaster protection:

By automating many of the manual tasks required to configure and monitor an Oracle Data Guard configuration, the broker enhances the high availability, data protection, and disaster protection capabilities that are inherent in Oracle Data Guard. Access is possible through a client to any system in the Oracle Data Guard configuration, eliminating any single point of failure. If the primary database fails, the broker automates the process for any one of the standby databases to replace the primary database and take over production processing. The database availability that Oracle Data Guard provides makes it easier to protect your data.

Higher availability and scalability with Oracle Real Application Clusters (Oracle RAC) Databases:

While Oracle Data Guard broker enhances disaster protection by maintaining transactionally consistent copies of the primary database, Oracle Data Guard, configured with Oracle high availability solutions such as Oracle Real Application Clusters (Oracle RAC) databases, further enhances the availability and scalability of any given copy of that database. The *intrasite* high availability of an Oracle RAC database complements the *intersite* protection that is provided by Oracle Data Guard broker.

For example, if one instance of an Oracle RAC primary database unexpectedly fails, Oracle Clusterware attempts to recover the failed instance and keep the primary database available. From an Oracle Data Guard perspective, the primary database remains available as long as at least one instance of the clustered database continues to transport redo data to the standby databases. If Oracle Clusterware is unable to recover a failed instance, the Oracle RAC database continues to run automatically with one less active instance. If the last instance of the primary database fails, the Oracle RAC database provides a way to fail over to a specified standby database. If the last instance of the primary database fails, and fast-start failover is enabled, the broker can

continue to provide high availability by automatically failing over to a pre-determined standby database.

The broker is integrated with Oracle Clusterware so that database role changes occur smoothly and seamlessly. This is especially apparent in the case of a planned role switchover (for example, when a physical standby database is directed to take over the primary role while the former primary database assumes the role of standby). The broker and Oracle Clusterware work together to temporarily suspend service availability on the primary database, accomplish the actual role change for both databases during which Oracle Clusterware works with the broker to properly restart the instances as necessary on the old primary database, and then start services defined on the new primary database. The broker manages the underlying Oracle Data Guard configuration and its database roles while Oracle Clusterware manages service availability that depends upon those roles. Applications that rely on Oracle Clusterware for managing service availability will see only a temporary suspension of service as the role change occurs in the Oracle Data Guard configuration.

Note that while Oracle Clusterware helps to maintain the availability of the individual instances of an Oracle RAC database, the broker coordinates actions that maintain one or more physical or logical copies of the database across multiple geographically dispersed locations to provide disaster protection. Together, the broker and Oracle Clusterware provide a strong foundation for Oracle's high-availability architecture.

 **See Also:**

Oracle Real Application Clusters Administration and Deployment Guide for information about Oracle Clusterware

Support for Oracle RAC One Node:

Oracle Data Guard and Oracle Data Guard broker are fully integrated with integrated with Oracle Real Application Clusters One Node (Oracle RAC One Node).

Automated creation of an Oracle Data Guard configuration:

The broker helps you to logically define and create an Oracle Data Guard configuration. The broker automatically communicates between the members of an Oracle Data Guard configuration using Oracle Net Services. The members can be local or remote, connected by a LAN or geographically dispersed over a WAN.

Cloud Control provides a wizard that automates the complex tasks involved in creating a broker configuration, including:

- Adding an existing standby database, or a new standby database created from existing backups taken through Cloud Control
- Configuring the standby control file, server parameter file, and datafiles
- Initializing communication with the standbys
- Creating standby redo log files
- Enabling Flashback Database if you plan to use fast-start failover

Although DGMGRL cannot automatically create a new standby, you can use DGMGRL commands to configure and monitor an existing standby, including those created using Cloud Control.

Easy configuration of additional standbys:

After you create an Oracle Data Guard configuration, you can add new or existing standbys to each Oracle Data Guard configuration. Cloud Control provides an Add Standby Database wizard to guide you through the process of adding more databases.

Simplified, centralized, and extended management:

You can issue commands to manage many aspects of the broker configuration. These include:

- Simplify the management of all components of the configuration, including the primary database, standbys, far sync instances, Zero Data Loss Recovery Appliances, redo transport services, and log apply services.
- Coordinate database state transitions and update member properties dynamically, with the broker recording the changes in a broker configuration file that includes information about all the members in the configuration. The broker propagates the changes to all databases in the configuration and their server parameter files.
- Simplify the control of the configuration protection modes (to maximize protection, to maximize availability, or to maximize performance).
- Invoke the Cloud Control verify operation to ensure that redo transport services and log apply services are configured and functioning properly.

Simplified switchover and failover operations:

The broker simplifies switchovers and failovers by allowing you to invoke them using a single key click in Cloud Control or a single command at the DGMGRL command-line interface (referred to in this documentation as *manual failover*). For *lights-out administration*, you can enable fast-start failover to allow the broker to determine if a failover is necessary and to initiate the failover to a pre-specified target standby automatically, with no need for DBA intervention. Fast-start failover can be configured to occur with no data loss or with a configurable amount of data loss.

Fast-start failover allows you to increase availability with less need for manual intervention, thereby reducing management costs. Manual failover gives you control over exactly when a failover occurs and to which target standby. Regardless of the method you choose, the broker coordinates the role transition on all databases in the configuration. Once failover is complete, the broker publishes a Fast Application Notification (FAN) event to notify applications that the new primary is available.

Note that you can use the `DBMS_DG` PL/SQL package to enable an application to initiate a fast-start failover when it encounters specific conditions. See [Application Initiated Fast-Start Failover](#) for more information.

Only one command is required to initiate complex role changes for switchover or failover operations across all databases in the configuration. The broker automates switchover and failover to a specified standby database in the broker configuration. Cloud Control enables you to select a new primary database from a set of viable standby databases (enabled and running, with normal status). The DGMGRL `SWITCHOVER` and `FAILOVER` commands only require you to specify the target standby database before automatically initiating and completing the many steps in switchover or failover operations across the multiple databases in the configuration.

Built-in monitoring and alert and control mechanisms:

The broker provides built-in validation that monitors the *health* of all of the databases in the configuration. While connected to any database in the configuration, you can capture diagnostic information and detect obvious and subtle problems quickly with centralized monitoring, testing, and performance tools. Both Cloud Control and DGMGRL retrieve a complete configuration view of the progress of redo transport services on the primary database and the progress of Redo Apply or SQL Apply on the standby database.

The ability to monitor local and remote databases and respond to events is significantly enhanced by the broker's health check mechanism and tight integration with the Cloud Control event management system.

Transparent to application:

Use of the broker is possible for any database because the broker works transparently with applications; no application code changes are required to accommodate a configuration that you manage with the broker.

Oracle Data Guard Broker Components

These are the components that make up Oracle Data Guard broker.

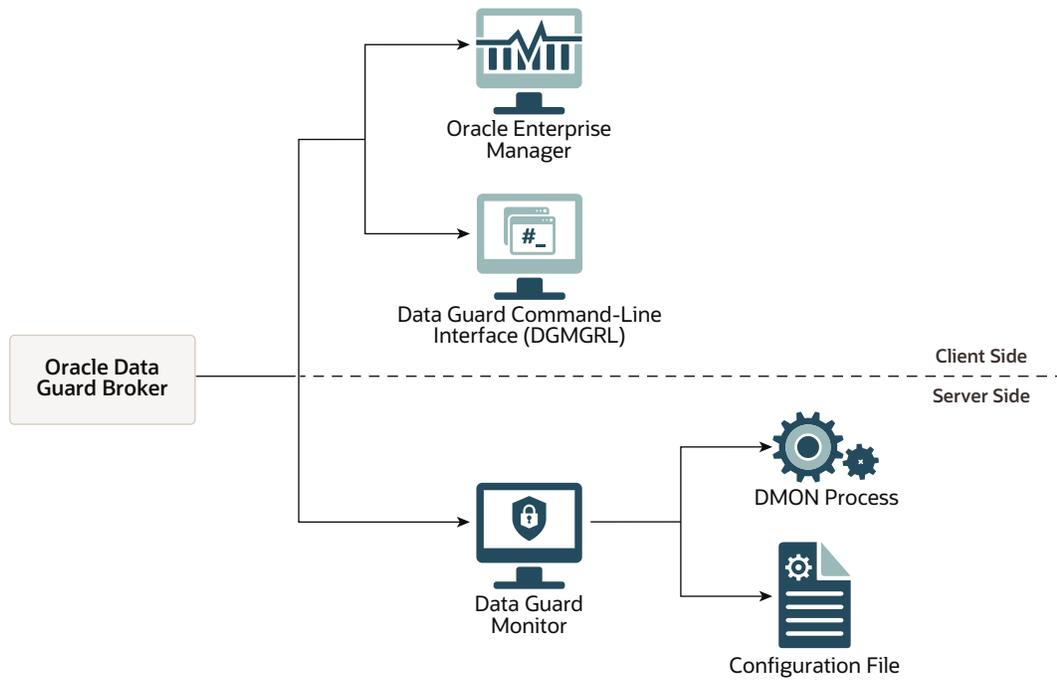
- [Oracle Enterprise Manager Cloud Control](#)
- [Oracle Data Guard Command-Line Interface \(DGMGRL\)](#)
- [Oracle Data Guard Monitor](#)

Cloud Control and the Oracle Data Guard command-line interface (DGMGRL) are the broker client interfaces that help you define and manage a configuration consisting of a collection of primary and standby databases, far sync instances, and Zero Data Loss Recovery Appliances. DGMGRL also includes commands to create **observer** processes for fast-start failover.

The Oracle Data Guard monitor is the broker server-side component that is integrated with the Oracle database. The Oracle Data Guard monitor is composed of several processes, including the DMON process, and broker configuration files that allow you to control the databases of that configuration, modify their behavior at runtime, monitor the overall health of the configuration, and provide notification of other operational characteristics.

[Figure 1-1](#) shows these components of the broker.

Figure 1-1 Oracle Data Guard Broker



Oracle Data Guard Broker User Interfaces

You can use either of the broker's user interfaces to create a broker configuration and to control and monitor the configuration.

The following sections describe the broker's user interfaces:

- [Oracle Enterprise Manager Cloud Control](#)
- [Oracle Data Guard Command-Line Interface \(DGMGRL\)](#)

Oracle Enterprise Manager Cloud Control

Oracle Enterprise Manager Cloud Control (Cloud Control) works with the Oracle Data Guard monitor to automate and simplify the management of an Oracle Data Guard configuration.

With Cloud Control, the complex operations of creating and managing standby databases are simplified through Oracle Data Guard management pages and wizards, including:

- An Add Standby Database wizard that helps you to create a broker configuration, if one does not already exist, having a primary database and a local or remote standby database. The wizard can create a physical, snapshot, or logical standby database or import an existing physical, snapshot, or logical (Oracle RAC or non-Oracle RAC) standby database. If the wizard creates a physical, snapshot, or logical standby database, the wizard also automates the creation of the standby control file, server parameter file, online and standby redo log files, and the standby datafiles.

- A switchover operation that helps you switch roles between the primary database and a standby database.
- A failover operation that changes one of the standby databases to the role of a primary database.
- Performance tools and graphs that help you monitor and tune redo transport services and log apply services.
- Property pages that allow you to set database properties on any database and, if applicable, the settings are immediately propagated to all other databases and server parameter files in the configuration.
- Event reporting through e-mail.

In addition, it makes all Oracle Net Services configuration changes necessary to support redo transport services and log apply services.

 **See Also:**

My Oracle Support note 787461.1 at <http://support.oracle.com> for information about which versions of Cloud Control are required to manage the full set of Oracle Data Guard features in various Oracle Database releases

Oracle Data Guard Command-Line Interface (DGMGRL)

The Oracle Data Guard command-line interface (DGMGRL) enables you to control and monitor an Oracle Data Guard configuration from the DGMGRL prompt or within scripts.

You can perform most of the activities required to manage and monitor the databases in the configuration using DGMGRL commands.

DGMGRL also includes commands to create observer processes that continuously monitor the primary and target standby databases and evaluate whether failover is necessary, and then initiate a fast-start failover when conditions warrant.

 **See Also:**

[Oracle Data Guard Command-Line Interface Reference](#) for complete reference information for the Oracle Data Guard command-line interface.

Oracle Data Guard Monitor

The configuration, control, and monitoring functions of the broker are implemented by server-side software and configuration files that are maintained for each database that the broker manages.

The software is called the Oracle Data Guard monitor. The following topics describe how the Oracle Data Guard monitor interacts with the Oracle database and with remote Oracle Data Guard monitors to manage the broker configuration.

Oracle Data Guard Monitor (DMON) Process

The Oracle Data Guard monitor process (DMON) is an Oracle background process that runs on every database instance that is managed by the broker.

When you start the Oracle Data Guard broker, a DMON process is created.



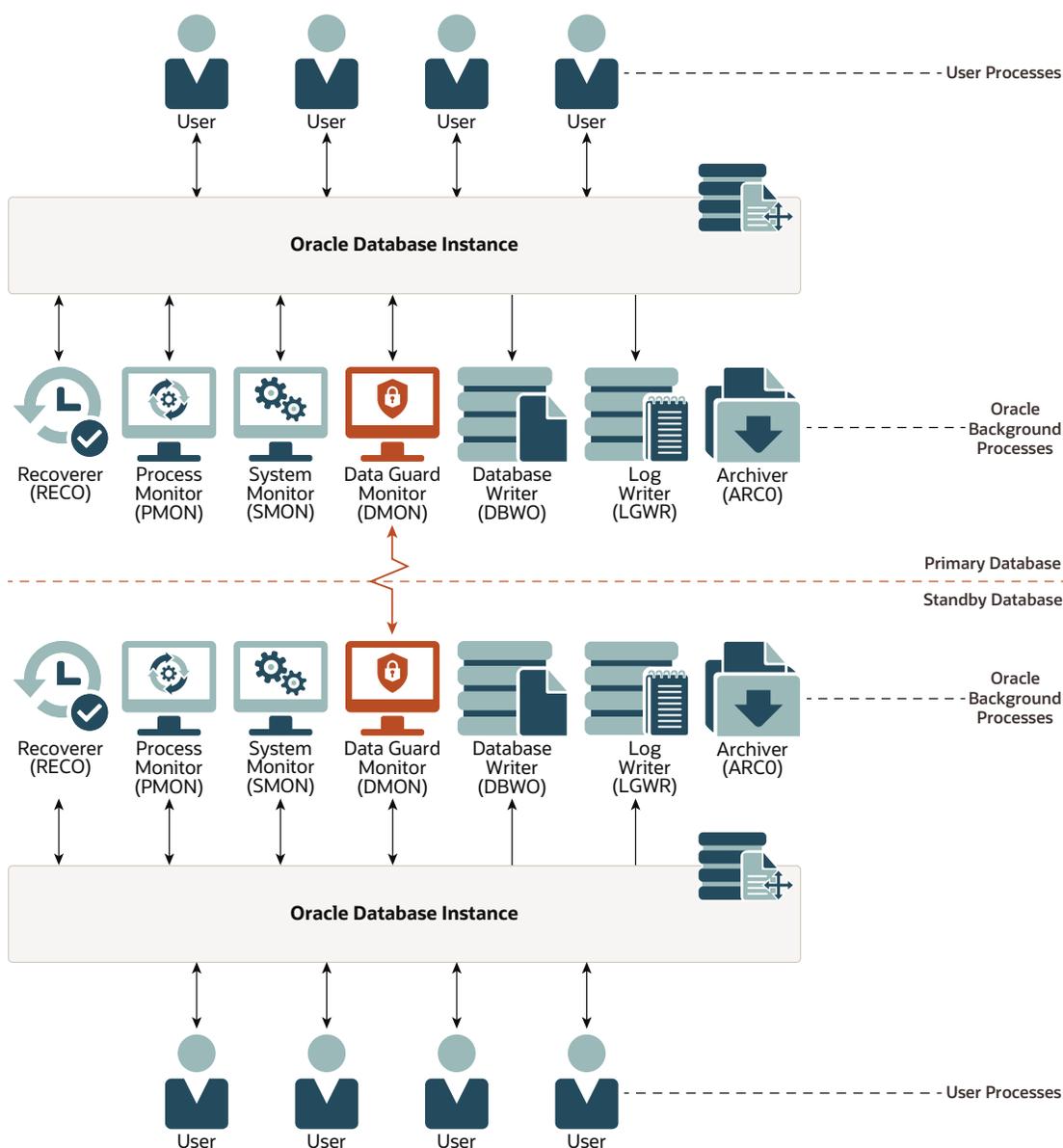
See Also:

[Starting the Data Guard Broker](#) for information on starting the broker

Whether you use Cloud Control or DGMGRL to manage a database, the DMON process is the server-side component that interacts with the local database and the DMON processes of the other databases to perform the requested function. The DMON process is also responsible for monitoring the health of the broker configuration and for ensuring that every database has a consistent description of the configuration.

[Figure 1-2](#) shows the broker's DMON process as one of several background processes that constitute an instance of the Oracle database. Each database instance shown in the figure has its own DMON process.

Figure 1-2 Databases With Broker (DMON) Processes



The zigzag arrow in the center of [Figure 1-2](#) represents the two-way Oracle Net Services communication channel that exists between the DMON processes of two databases in the same broker configuration.

This two-way communication channel is used to pass requests between databases and to monitor the health of all of the databases in the broker configuration.

Monitoring Data Guard Broker Processes

Use the `V$DATAGUARD_PROCESS` view to monitor the status of broker processes, as well as other Data Guard processes. This view contains a row showing information for the broker DMON process. The following are the columns and values shown for the DMON process (other columns in the view are generally not relevant to the broker):

- NAME — DMON
- ROLE — broker monitor
- PID — operating system process identifier of the process
- PROC_TIME — timestamp of when the process was started or registered for inclusion in the view

 **See Also:**

- *Oracle Database Reference* for more information about the `V$DATAGUARD_PROCESS` view

Configuration Management

The broker's DMON process persistently maintains information about all members of the broker configuration in a binary configuration file.

A copy of the configuration file is maintained by the DMON process for each of the databases that belong to the broker configuration. For an Oracle RAC database, all instances of the database share the same configuration file.

This configuration file contains information that describes the states and properties of the databases in the configuration. For example, the file records the databases that are part of the configuration, the roles and properties of each of the databases, and the state of each database in the configuration.

The configuration data is managed transparently by the DMON process to ensure that the configuration information is kept consistent across all of the databases. The broker uses the data in the configuration file to configure and start the databases, control each database's behavior, and provide information to DGMGRL and Cloud Control.

 **See Also:**

[Managing Database Properties](#) for more information

Whenever you add databases to a broker configuration, or make a change to an existing database's properties, each DMON process records the new information in its copy of the configuration file.

Database Property Management

The broker uses various properties associated with each database to control the database's behavior. The properties are recorded in the configuration file.

To ensure that the broker can update the values of parameters in both the database itself and in the configuration file, you must use a server parameter file to control static and dynamic initialization parameters. The use of a server parameter file gives the broker a mechanism that allows it to reconcile property values selected by the

database administrator (DBA) when using the broker with any related initialization parameter values recorded in the server parameter file.

When you set values for database properties in the broker configuration, the broker records the change in the configuration file and propagates the change to all of the databases in the Oracle Data Guard configuration.

 **Note:**

The broker supports both the default and nondefault server parameter file filenames. If you use a nondefault server parameter filename, the initialization parameter file must include the complete filename and location of the server parameter file. If this is an Oracle RAC database, there must be one nondefault server parameter file for all instances.

 **See Also:**

[Configurable \(Changeable\) Properties](#) for more information

2

Oracle Data Guard Installation

For all members that you want to add to a broker configuration, you must complete installation and configuration tasks.

- [Oracle Data Guard Installation](#)
- [Prerequisites](#)

See Also:

[Oracle Data Guard Broker Upgrading and Downgrading](#) for help with upgrading to a new release in an Oracle Data Guard broker configuration

Oracle Data Guard Installation

Oracle Data Guard is included with the Enterprise Edition and Personal Edition of the Oracle database software.

You can manage an Oracle Data Guard configuration by using either SQL*Plus, the Oracle Data Guard broker's command-line interface (DGMGRL), or a compatible version of Oracle Enterprise Manager Cloud Control (Cloud Control).

Install the Oracle Enterprise Edition or Personal Edition database software on each location you expect to include in broker configurations. You must install a compatible version of Cloud Control if you want to use the graphical user interface for Oracle Data Guard.

In addition, to use fast-start failover you must install DGMGRL and run the observer software. Oracle recommends running observers on computer systems that are separate from the primary and standby systems. To install DGMGRL on an observer computer, use one of the methods described in the following list:

- Install the complete Oracle Client Administrator by choosing the `Administrator` option from Oracle Universal Installer.

This installation includes DGMGRL but it does not include the Cloud Control agent. This allows you to manage observer processes using DGMGRL commands, but not Cloud Control. (The Cloud Control agent can be installed separately.)

- Install the full Oracle Database Enterprise Edition or Personal Edition software kit

This installation includes DGMGRL and the Cloud Control agent, allowing you to manage observer processes using Cloud Control or DGMGRL commands.

Note:

An observer can be run from any platform that supports it, and that platform can be different from the platform of the primary or target standby database.

**Note:**

Zero Data Loss Recovery Appliance does not require broker software to be installed.

Prerequisites

These are the conditions that must be met before you can use the broker.

- The primary and standby databases and far sync instances must be using the same version of Oracle Database and each can be installed in either a single-instance or multi-instance environment. The database must be licensed for Oracle Enterprise Edition or Personal Edition.
- You must use a server parameter file (SPFILE) to ensure the broker can persistently reconcile values between broker properties and any related initialization parameter values. See [Configurable \(Changeable\) Properties](#) for more information.

If any of the databases in the configuration is an Oracle RAC database, you must configure the server parameter file (SPFILE) appropriately for use in an Oracle RAC environment.

**See Also:**

Oracle Real Application Clusters Administration and Deployment Guide for information about initialization files in an Oracle RAC

- The value of the `DG_BROKER_START` initialization parameter must be set to `TRUE` for all databases in the configuration. See [Starting the Data Guard Broker](#) for more information. (Cloud Control sets this parameter automatically.)
- If any of the databases in the configuration is an Oracle RAC database, you must set up the `DG_BROKER_CONFIG_FILEn` initialization parameters for that database such that they point to the same shared files for all instances of that database. The shared files could be files on a cluster file system, if available, or stored using Oracle Automatic Storage Management (Oracle ASM).

**See Also:**

Configuration file information in [Configuration Management](#). Also, see [Setting Up the Broker Configuration Files](#) for details about setting up the broker configuration file.

- Network configuration files must be set up on the primary database and on the standby database if you configure an existing standby database into the broker configuration. Cloud Control can assist you in creating the configuration files when creating a standby database.

 **See Also:**

Oracle Database Net Services Administrator's Guide for more information about network configuration files

- In a broker configuration, you use the `DGConnectIdentifier` property to specify a connect identifier for each database. The connect identifier for a database must:
 - Allow all other databases in the configuration to reach it.
 - Allow all instances of an Oracle RAC database to be reached.
 - Specify a service that all instances dynamically register with the listeners so that connect-time failover on an Oracle RAC database is possible.

 **Caution:**

The service should NOT be one that is defined and managed by Oracle Clusterware.

- Have failover attributes set to allow the primary database's Redo Transport Services to continue shipping redo data to an Oracle RAC standby database, even if the receiving instance of that standby database has failed.

 **See Also:**

Oracle Database Net Services Administrator's Guide for more information about connect identifiers

- To enable DGMGRL to restart instances during the course of broker operations, a static service must be registered with the local listener of each instance. For configurations where Oracle Clusterware or Oracle Restart are not being used, a static service must be registered with the local listener associated with each instance. The static service allows DGMGRL to restart instances during the course of broker operations and to allows the observer to restart instances during automatic reinstatement of the old primary database after a fast-start failover has occurred.

By default, the broker assumes a static service name of `db_unique_name_DGMGRL.db_domain` and expects the listener has been started with the following content in the `listener.ora` file:

```
LISTENER = (DESCRIPTION =
  (ADDRESS_LIST=(ADDRESS=(PROTOCOL=tcp) (HOST=host_name)
    (PORT=port_num))))
SID_LIST_LISTENER=(SID_LIST=(SID_DESC=(SID_NAME=sid_name)
  (GLOBAL_DBNAME=db_unique_name_DGMGRL.db_domain)
  (ORACLE_HOME=oracle_home)
  (ENVS="TNS_ADMIN=oracle_home/network/admin")))
```

Alternatively, you can use a different static service name. If you do, be sure to modify the `StaticConnectIdentifier` instance-specific property to reflect the static service that has been registered.

To ensure that the connect identifier has been set up correctly, use the `VALIDATE STATIC CONNECT IDENTIFIER` command.

 **See Also:**

- See *Oracle Database Net Services Administrator's Guide* for more information about configuring static services
 - See the My Oracle Support Note 1387859.1 at <http://support.oracle.com> for additional information about using static services and the `StaticConnectIdentifier` configurable property
 - [VALIDATE STATIC CONNECT IDENTIFIER](#)
- The primary database must be opened in `ARCHIVELOG` mode.
 - You must set the `COMPATIBLE` initialization parameter to 12.2.0 or higher for both the primary and standby databases. However, if you want to take advantage of Oracle Database Release 21c new features, then set the `COMPATIBLE` parameter to 21.1 on all databases within the Oracle Data Guard configuration.

For the broker to work, the `COMPATIBLE` initialization parameter must be set to the same value on both the primary and standby databases. If the values differ, redo transport services may be unable to transmit redo data from the primary database to the standby databases.
 - Select and configure a redo transport authentication method, as described in *Oracle Data Guard Concepts and Administration*.

 **See Also:**

- [Starting the Data Guard Broker](#) for more information about preparing and starting Oracle Data Guard broker

3

Managing Broker Configurations

Use Oracle Data Guard broker to manage and monitor the configuration. Oracle Enterprise Manager uses the broker to report health and other operational characteristics.

- [Configuration Support](#)
- [Configuration Properties](#)
- [Setting Up the Broker Configuration Files](#)
- [Starting the Data Guard Broker](#)
- [Management Cycle of a Broker Configuration](#)
- [Enable and Disable Operations](#)
- [Configuration Status](#)

Overview of Broker Configurations

Data Guard broker provides support for two mutually-exclusive configurations.

- **Data Guard for Container Databases (DG CDB)**
Protects the whole primary database. A role change converts the whole database from a primary database to a standby database. The standby database includes all the pluggable databases (PDBs) within the primary database. Deploy one or more physical or logical standby databases to protect the redo stream of the primary database.
- **Data Guard for Pluggable Databases (DG PDB)**
Protects one or more PDBs in a primary database. The destination for redo data is another primary database, referred to as a target database. When there is a failure in a protected PDB, the broker can switchover or failover to the corresponding PDB in the target database. There is no need for a role change of the whole database.

Configuration Support

Oracle Data Guard broker enables you to create a broker configuration consisting of up to 253 members, including one primary database and a combination of standby databases, far sync instances, and Zero Data Loss Recovery Appliances.

The broker controls the members of the configuration, monitors the health of the configuration, and reports health and other operational characteristics through the Oracle Enterprise Management notification mechanisms if you are using Oracle Enterprise Manager Cloud Control, or through `SHOW` commands if you are using DGMGRL.

A supported Oracle Data Guard configuration contains the following components:

- A primary database (Oracle RAC or non-Oracle RAC)
- A combination of standby databases, far sync instances, and recovery appliances that receive redo from the primary database.

- Oracle Net Services network configuration that defines a connection between the databases
- Standby (archived redo log files) destination parameters and configuration properties
- Redo transport services that transmit the redo data from the primary database to the standbys and far sync instances
- Log apply services that apply redo data to the standby databases from the archived redo log files or standby redo log files

The Oracle Data Guard log apply services update standby databases with redo data that is transmitted automatically from the primary database by redo transport services. The archived redo log files and standby redo log files contain all of the database changes except for *unrecoverable* or *unlogged* changes.

- On physical standby databases, Redo Apply applies the redo data to keep the standby consistent with the primary database.
- On logical standby databases, SQL Apply applies the redo data to keep the standby consistent with the primary database.
- On snapshot standby databases, the redo data is received but not applied until the snapshot standby database is converted back to a physical standby database.
- On far sync instances, the redo data is received and then forwarded to a physical standby database. A far sync instance does not have data files and does not apply any of the redo data it has received.

The broker's Oracle Data Guard monitor (DMON) process configures and maintains the broker configuration as a group of objects that you can manage and monitor as a single unit. Thus, when you enter a command that affects multiple databases, the DMON process:

- Carries out your request on the primary database
- Coordinates with the DMON process for each of the other databases, as required for your request
- Updates the configuration file on the local system
- Communicates with the DMON process for each of the other databases to update their copies of the configuration file

Through the DMON process, you can configure, monitor, and control the databases and the configuration together as a unit. If you disable the configuration, broker management of all of the databases in the configuration is also disabled. If you later enable the configuration, broker management is enabled for each database in the configuration.

Figure 3-1 shows a broker configuration with a primary database and physical standby database.

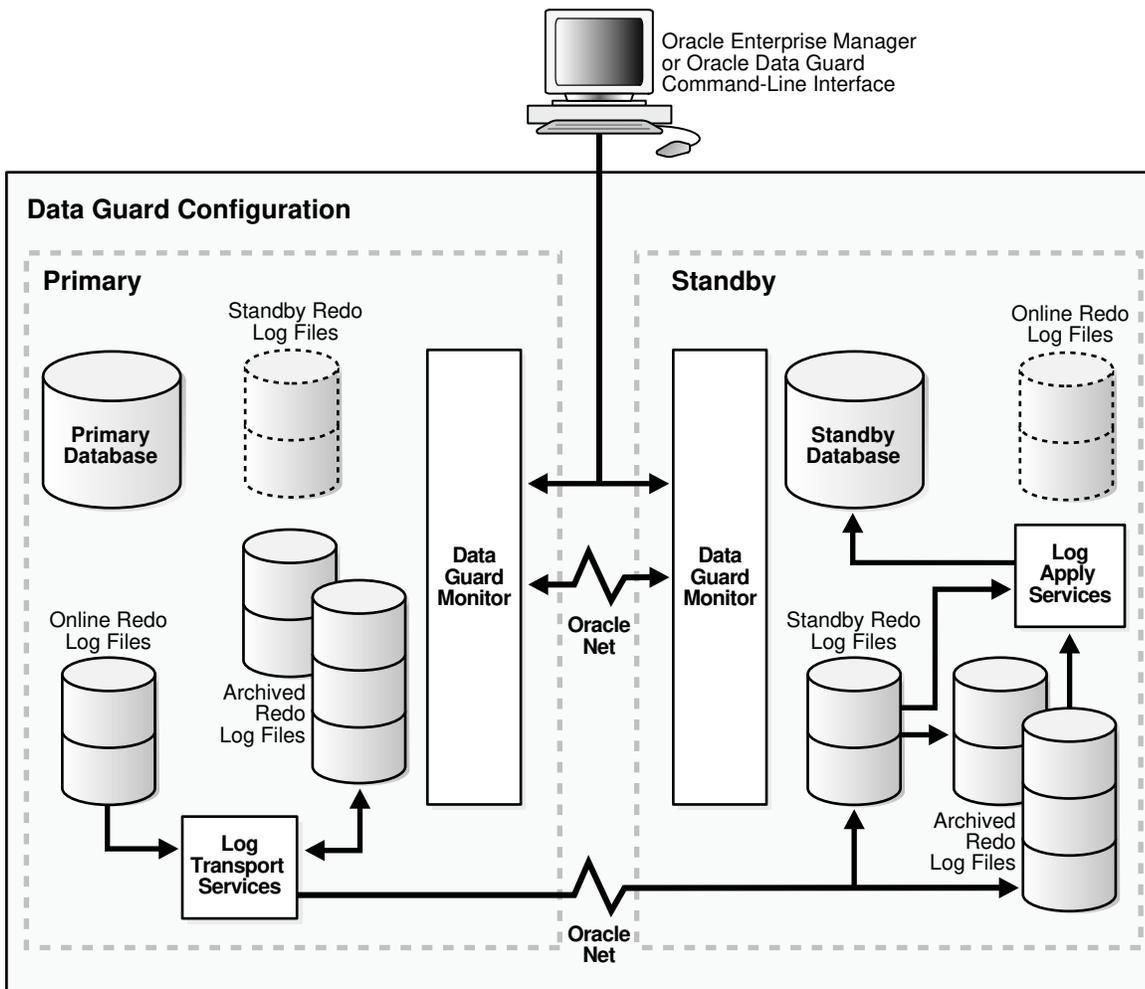
On the primary database, the figure shows the redo transport services in addition to the following main components: the primary database, DMON, the online redo log files, and the archived redo log files. The figure also shows standby redo log files in outline form on the primary side; the standby redo logs are outlined to indicate they are currently inactive but have been configured in preparation for a switchover or failover to the standby role. The physical standby database includes the following components: a standby database, log apply services, DMON, archived redo log files, and standby redo log files. The online redo log files on the physical standby database are outlined

to indicate they are currently inactive but have been configured in preparation for a switchover or failover to the primary role.

 **See Also:**

Oracle Data Guard Concepts and Administration for more information about standby databases

Figure 3-1 Oracle Data Guard Broker Configuration



Configuration-Wide Service Names

The broker publishes a service, using the same name, on each configuration member. The default name of this configuration-wide service is *primarydbname_CFG*, where a suffix of *_CFG* is appended to the value of the *DB_UNIQUE_NAME* initialization parameter of the primary database at the time the configuration is created. To change the name of the configuration-wide service, use the *ConfigurationWideServiceName* property. See [ConfigurationWideServiceName](#).

Configuration Support for DG PDB

Two primary databases are required to provide support for Data Guard for individual pluggable databases. Each primary database constitutes its own broker configuration, thus requiring two broker configurations to provide data protection. Communication between the two configurations is established using Oracle Net.

The broker provides support for DG PDB through a set of new DGMGRL commands that will be described later.

DG PDB Terminology

Learn about the terminology used with DG PDB configurations.

- **Source PDB**
A pluggable database (PDB) within a container database that must be protected. Data Guard maintains a copy of the source PDB in the target container database.
- **Source database**
A primary container database that contains one or more source PDBs, and is the source of redo data.
- **Target PDB**
A PDB that is a copy of the source PDB. Redo data from the source PDB is applied to the target PDB. If there is a failure in the source PDB, the broker can switchover or failover to the target PDB.
- **Target database**
A primary container database that contains one or more target PDBs. It instantiates and maintains the source PDBs that must be protected.
- **Source configuration**
A Data Guard broker configuration that contains the source database.
- **Target configuration**
The Data Guard broker configuration that contains the target database.

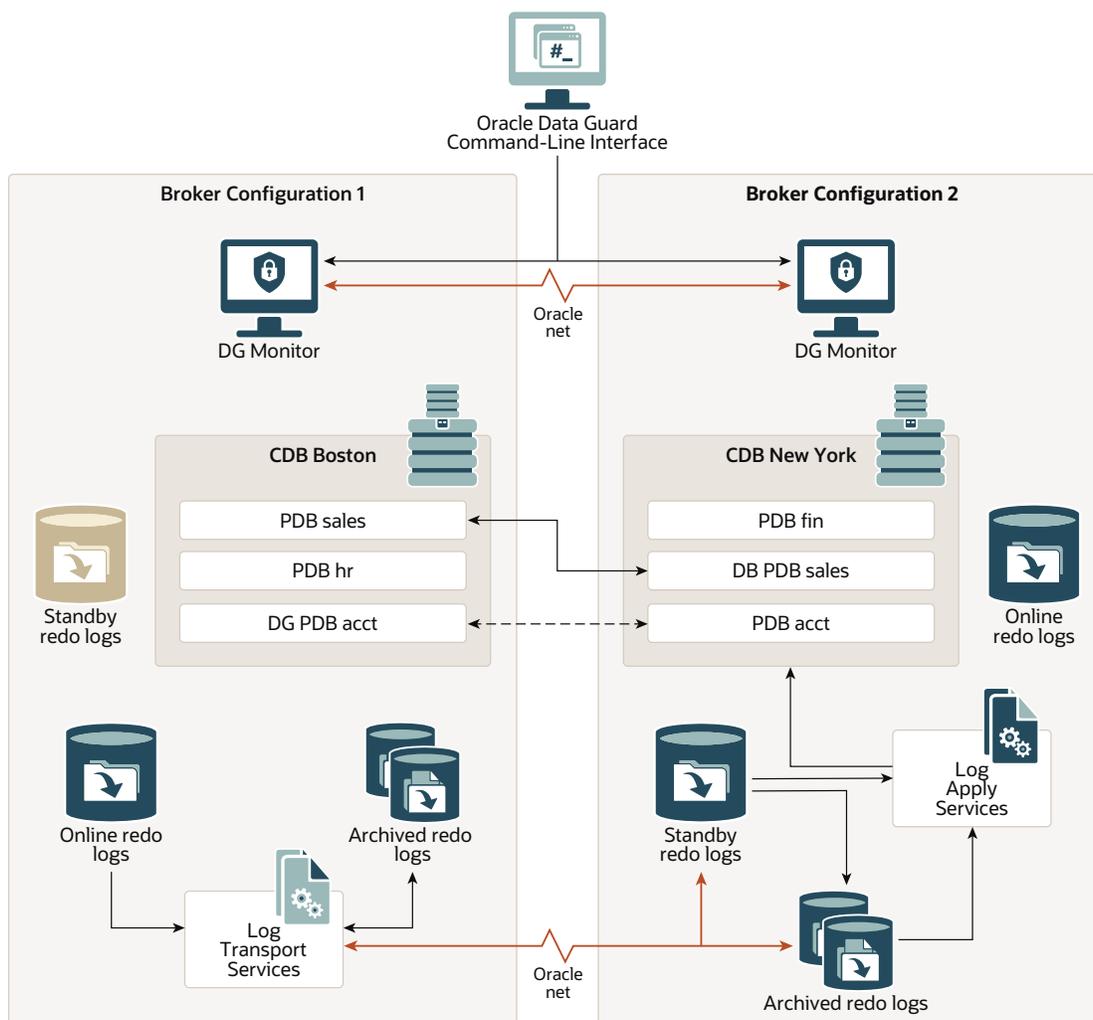
Components of DG PDB Configuration

A DG PDB configuration must contain 2 primary databases, each within their own configuration.

One or more source PDBs from one container database can be set up for data guard protection at the other container database. A container database may contain one or more source, target, or both source and target PDBs. Redo is shipped from the container database that contains source PDBs, to the container database that contains the target PDBs. Since a container database could contain both source and target PDBs, it could simultaneously be shipping redo, as well as receiving redo from the other container database.

[Figure 3-2](#) illustrates the components of a DG PDB configuration.

Figure 3-2 Oracle DG PDB Configuration



Broker Configuration 1 is the source configuration PDB sales and a target configuration for DG PDB acct. Broker Configuration 2 is a source configuration for PDB acct and a target configuration for DG PDB sales. Both configurations show the redo transport services in addition to the following main components: the source DMON, the online redo log files, standby redo log files, and the archived redo log files. The source PDB `pdb_sales` is instantiated as `dgpdb_sales` in the target database `cdb_newyork`.

The primary database in a DG PDB configuration can be either a source or target of redo data. Depending upon where the source and target PDBs reside, a primary CDB can be both a source and a target CDB. There can be a maximum of two CDBs. For example, in the above figure, `cdb_newyork` contains a PDB named `pdb_acct` which is connected using a dotted line to `dgpdb_acct` in the database `cdb_boston`. This indicates that the source PDB `pdb_acct` has a DG PDB instantiated as `dgpdb_acct`. However, a primary database can be the target for only one redo stream, that is, only one source database.

The Oracle Data Guard monitor (DMON) process configures and maintains the broker configuration as a group of objects that you can manage and monitor as a single unit. The functionality is the same as that described in a regular broker configuration.

About the DGPDB_INT User Account

The `DGPDB_INT` user account is used by the database server when making connections to other sites involved in the DG PDB configuration. Typically this occurs during creation and switchover of a standby PDB.

When the first DG PDB is added, DGMGRL asks for the password of the `DGPDB_INT` account. If the account is still locked at the remote site, DGMGRL unlocks it, assigns to it the password supplied, and grants the required privileges to it. The password is securely recorded using the `DBMS_CREDENTIAL` package at the local site. Along with the connect identifier for the remote site, the credential and the `DGPDB_INT` account are used to make connections to the remote site when required.

Before you can create a target PDB for a source PDB, the `SYSDBA` administrative privilege must be granted to the `DGPDB_INT` user in the source PDB. DGMGRL makes this grant as part of creating a target PDB for a given source PDB.

DGMGRL revokes the privileges and locks the `DGPDB_INT` account when appropriate.

DG PDB Configuration Restrictions

Several restrictions apply when using the DG PDB configuration.

A DG PDB configuration does not support the following:

- Snapshot standby databases, far sync instances, and bystander standbys
- Maximum availability and maximum protection modes
- Oracle Active Data Guard features
- Rolling upgrades using the `DBMS_ROLLING` package
- Multiple databases in the target configuration

A target database can receive and apply redo from one source database only.

- Oracle GoldenGate as part of a configuration that provides support for DG PDB configurations
- Downstream data capture for Oracle GoldenGate
- Data Guard broker external destinations
- Data Guard broker functionality for Zero Data Loss Recovery Appliance (ZDLRA)
- Backups to ZDLRA
- Application containers

Only individual PDBs are supported for Oracle Data Guard.

Workflow for Using DG PDB Configuration

Use the workflow to understand the tasks required to set up and manage a DG PDB configuration.

To provide data protection for one or more PDBs using the DG PDB configuration:

1. Identify the source PDBs that must be protected. These source PDBs are contained in the source database.

2. Identify the target database that is the destination for redo data from the source database.
3. Establish a relationship between the source configuration and the target configuration. See [Scenario 2: Establish a Connection Between the Configurations](#).
4. Set up standby redo log and archived redo log locations in the target configuration. See [Specifying Locations for Archived Redo Log Files](#).
5. Instantiate the source PDB at the target database. See [Scenario 3: Setting Up Data Guard for the Source PDB](#).
6. Start redo transport from the source database to the target database and start recovery at the target database. See [Scenario 4: Enable the DG PDB Configuration and Start Redo Transport](#).
7. When required, perform a switchover operation. See [Performing PDB Switchover](#).
Or, when required, you initiate a failover to the target PDB. See [Performing PDB Failover](#).
8. Monitor the progress of redo transport at the source database and recovery at the target database. See [Scenario 7: Monitoring a DG PDB Configuration](#).

Configuration Properties

Configuration properties control the behavior of a broker configuration.

You can view and dynamically update the values of these properties using either DGMGRL or Cloud Control. However, some properties can only be updated through DGMGRL.

A configuration property has configuration-wide scope; meaning that the value you set for the property applies uniformly to each database in the configuration.

Note:

Starting with Oracle Database Release 19c, properties related to database initialization parameters, except `log_archive_dest_n` and `log_archive_dest_state_n`, are no longer stored in the broker configuration file.

See Also:

[Oracle Data Guard Broker Properties](#) for complete descriptions of all broker configuration properties

Setting Up the Broker Configuration Files

When the broker is started for the first time, configuration files are automatically created and named using a default path name and filename that is operating-system specific.

Two copies of the configuration file are maintained for each database so as to always have a record of the last known *valid* state of the configuration. You can override the default path name and filename by setting the following initialization parameters for that database:

```
DG_BROKER_CONFIG_FILE1  
DG_BROKER_CONFIG_FILE2
```

Note the following restrictions when setting the `DG_BROKER_CONFIG_FILE1` and `DG_BROKER_CONFIG_FILE2` initialization parameters:

- These parameters can only be set or changed when the Oracle Data Guard broker is not running (`DG_BROKER_START=FALSE`).
- These parameters must specify an Oracle ASM, Oracle OCFS, or NFS that is shared for all Oracle RAC instances.
- When an upgrade is performed using the PL/SQL package, `DBMS_ROLLING`, the `DG_BROKER_CONFIG_FILEn` parameters must specify a location outside of Oracle home. (The default location is the `db` directory in Oracle home.)

The Oracle Data Guard broker works with databases that use either Oracle managed or user managed datafiles. These datafiles can reside on a file system or an Oracle ASM disk group. The following section contains these topics:

- [Renaming the Broker Configuration Files](#)
- [Managing Broker Configuration Files in an Oracle RAC Environment](#)

Renaming the Broker Configuration Files

You can change the names of the broker configuration files by issuing the `ALTER SYSTEM SQL` statement.

However, you cannot alter these parameters when the broker's `DMON` process is running. To change the names of configuration files for a given database, perform the following steps:

1. Disable the broker configuration using the `DGMGRL DISABLE` command. See [Enable and Disable Operations](#).
2. Stop the Oracle Data Guard broker `DMON` process using the following SQL statement:

```
SQL> ALTER SYSTEM SET DG_BROKER_START=FALSE;
```

3. Change the configuration filenames for the database:

```
SQL> ALTER SYSTEM SET DG_BROKER_CONFIG_FILE1=filespec1;  
SQL> ALTER SYSTEM SET DG_BROKER_CONFIG_FILE2=filespec2;
```

Note:

If the broker is managing an Oracle RAC database, the value of `DG_BROKER_CONFIG_FILE1` and the value of `DG_BROKER_CONFIG_FILE2` for each of the instances must point to the same set of physical files.

4. The method of moving the files depends upon where they currently reside and where you want to move them to:
 - If the files reside on an operating system file system, use operating system commands to move the files to their new location.

- If the old or new location is an Oracle ASM disk group, use the `DBMS_FILE_TRANSFER.COPY_FILE` function to transfer the files to their new location.
5. Restart the Oracle Data Guard broker DMON process, as follows:


```
SQL> ALTER SYSTEM SET DG_BROKER_START=TRUE;
```
 6. Enable the broker configuration using the `DGMGRL ENABLE` command or the Enable operation in the Oracle Data Guard management pages of Cloud Control.

Managing Broker Configuration Files in an Oracle RAC Environment

In an Oracle RAC environment, all instances of a database must reference the same set of configuration files.

This means that if the broker is managing an Oracle RAC database, then the value of `DG_BROKER_CONFIG_FILE1` and the value of `DG_BROKER_CONFIG_FILE2` for each of the instances must point to the same set of physical files. The configuration files can be deployed using one of the following methods:

- [Using Cluster File System \(CFS\) for Configuration Files](#)
- [Using Oracle ASM Disk Groups for Configuration Files](#)

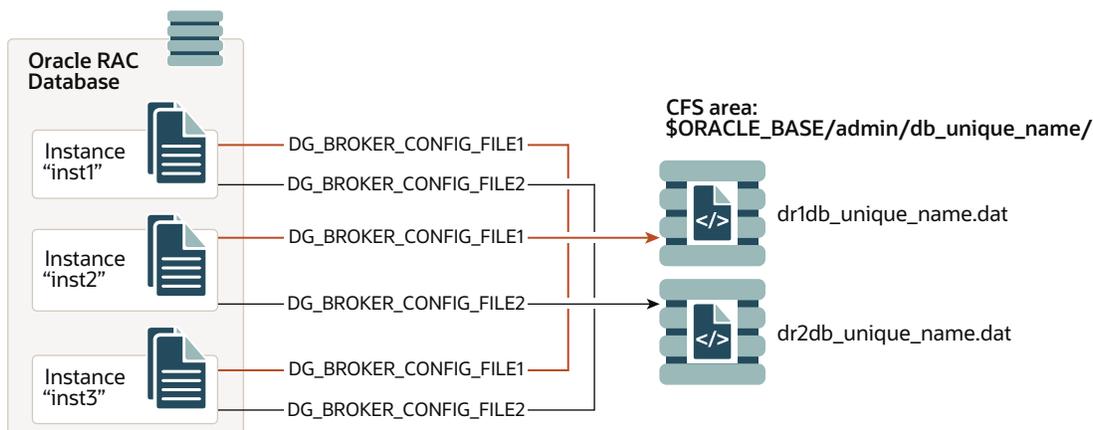
Using Cluster File System (CFS) for Configuration Files

The broker configuration files can reside on a cluster file system (CFS).

If they do, then , the `DG_BROKER_CONFIG_FILEn` parameters on all of the instances must be set to these files including the path to the CFS area. [Figure 3-3](#) shows the set up for the broker configuration files on CFS. In this scenario, the parameters and value for all instances would be:

```
DG_BROKER_CONFIG_FILE1=$ORACLE_BASE/admin/db_unique_name/dr1db_unique_name.dat
DG_BROKER_CONFIG_FILE2=$ORACLE_BASE/admin/db_unique_name/dr2db_unique_name.dat
```

Figure 3-3 Broker Configuration Setup in a CFS Area



Using Oracle ASM Disk Groups for Configuration Files

The broker's configuration files can also reside on an Oracle ASM disk group.

Use one of the following techniques to set the location of broker configuration files:

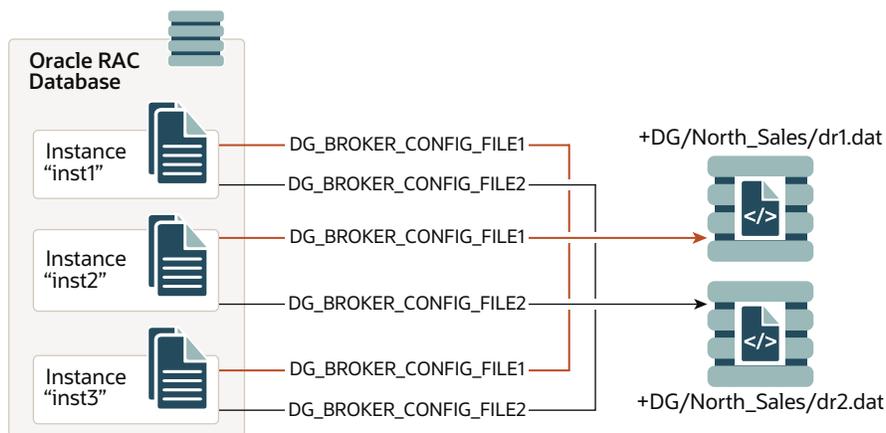
- Specify the name of the configuration file along with the complete path
Set the `DG_BROKER_CONFIG_FILE1` and `DG_BROKER_CONFIG_FILE2` initialization parameters to a string value that includes the name of an existing Oracle ASM disk group, an existing directory in that disk group, and the name of the configuration file itself.
- Specify only a disk group name
Set the `DG_BROKER_CONFIG_FILE1` and `DG_BROKER_CONFIG_FILE2` initialization parameters to the name of an ASM disk group. The broker automatically translates the disk group name to a predefined path and file name. The configuration files are as follows:

```
DG_BROKER_CONFIG_FILE1 = '+DG/db_unique_name/DATAGUARDCONFIG/
dr1db_unique_name.dat'
DG_BROKER_CONFIG_FILE2 = '+DG/db_unique_name/DATAGUARDCONFIG/
dr2db_unique_name.dat'
```

Figure 3-4 shows the setup for the broker configuration files on Oracle ASM devices. In this scenario, the parameters and values would be specified, as follows:

```
ALTER SYSTEM SET DG_BROKER_CONFIG_FILE1 = '+DG/North_Sales/dr1.dat' SCOPE=BOTH;
ALTER SYSTEM SET DG_BROKER_CONFIG_FILE2 = '+DG/North_Sales/dr2.dat' SCOPE=BOTH;
```

Figure 3-4 Broker Configuration Setup with Oracle ASM



Starting the Data Guard Broker

After setting up the configuration files, the `DG_BROKER_START` initialization parameter must be set to `TRUE` for each database to start the DMON processes.

By default, the `DG_BROKER_START` initialization parameter is set to `FALSE`. However, you can set the value in the following ways:

- If you are using Cloud Control, it automatically sets the `DG_BROKER_START` initialization parameter to `TRUE` for new standby databases that it creates.
- If you are using DGMGRL, you must explicitly set the `DG_BROKER_START` initialization parameter to `TRUE`; otherwise, the broker will not start. You can set the `DG_BROKER_START` initialization parameter with the following SQL statement:

```
SQL> ALTER SYSTEM SET DG_BROKER_START=TRUE;

System altered.

SQL> SHOW PARAMETER DG_BROKER_START

NAME                                TYPE        VALUE
-----                                -
dg_broker_start                      boolean     TRUE
```

Whether you use Cloud Control or DGMGRL, set the value of the `DG_BROKER_START` initialization parameter to `TRUE` on each database and far sync instance. Doing so ensures that Oracle Data Guard broker will start automatically the next time you start any instance of the database.

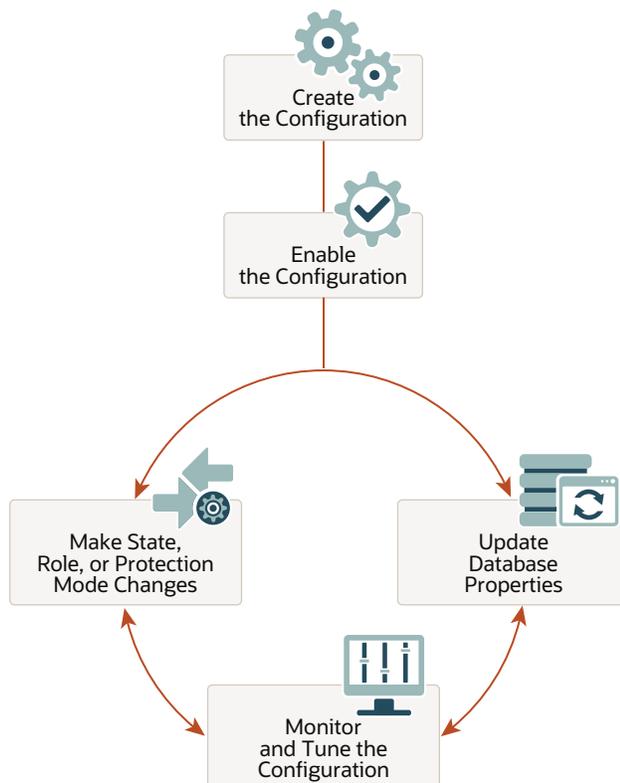
 **Note:**

If the initialization parameter `DG_BROKER_START=TRUE`, then archivelog mode is automatically enabled when the primary database is enabled and the database is not open on any instance.

Management Cycle of a Broker Configuration

This figure shows the life cycle of a broker configuration and the accompanying text explains each phase of the cycle.

Figure 3-5 Life Cycle of a Broker Configuration and Its Databases





Note:

For DG PDB configurations, you cannot make state changes or protection mode changes.

Create the Broker Configuration

When using Cloud Control, the Add Standby Database wizard can either add an existing single-instance or Oracle RAC standby database into the configuration or create a new single-instance or Oracle RAC standby database and add it to the configuration. The standby database can be a physical, logical, or snapshot database.

When using DGMGRL, the primary database and a standby database must already exist. You construct the standby database from backups of the primary database control files and datafiles, and then prepare it for recovery.

Enable the Broker Configuration

An Oracle Data Guard configuration must be enabled to be managed or monitored by the broker. Conversely, you disable a configuration if you no longer want to manage it with the broker. When you disable a configuration, broker management of all of its databases is also disabled.



Note:

You can enable or disable the configuration using DGMGRL. You cannot disable the configuration using Cloud Control. You can enable the configuration using Cloud Control in the event that it was previously disabled using DGMGRL.

A broker configuration, when first created using Cloud Control, is automatically enabled as soon as the Add Standby Database wizard completes.

A broker configuration, when first created using DGMGRL, is in a disabled condition. This means its constituent databases are not yet under active control of the broker. When you finish configuring the databases into a broker configuration with DGMGRL, you must enable the configuration to allow the broker to manage the configuration.

You can enable:

- The entire configuration, including all of its databases
- An individual standby database

You can easily disable a database if a problem occurs such that it cannot function properly in a broker configuration. Note that you cannot disable the primary database. You must disable the entire configuration to disable the primary database.

You may also want to disable a configuration temporarily, and then change some properties in the broker configuration without affecting the actual database properties. The changed properties will take effect when the configuration is enabled again for management by the broker.

Make Role Changes Within the Broker Configuration, As Needed

At any time, you can issue a single command to change the roles of the databases in the configuration. If some event renders the primary database unusable, you can fail over one of the standby databases to become the new primary database.

If Flashback Database was enabled on the former primary database, then after failover has completed, you can reinstate the former primary database as a standby database for the new primary database. This prevents you from having to re-create the old primary database from a copy of the new primary database.

In addition, planned downtime for maintenance can be reduced because you can quickly switch over production processing from the current primary database to a standby database, and then switch back again after the planned maintenance.



See Also:

[Switchover and Failover Operations](#) for more information about role changes

Convert to Snapshot Standby Database

At any time, you can issue a single command to convert a physical standby database to a snapshot standby database. The snapshot standby database is a fully updatable database that receives redo data generated from the primary database, but does not apply it.

Once you are done using the snapshot standby database, you can again issue a single command to convert it to a physical standby database. After the conversion to a physical standby has completed, Redo Apply services will start (assuming the state is `APPLY-ON`) and apply all of the accumulated redo data.

Make State Changes to the Databases, As Needed

When you enable a configuration for the first time, the broker, by default, starts redo transport services on the primary and starts log apply services on the standby (except for a snapshot standby).

At any time, you can issue a single command through Cloud Control or DGMGRL to change the state of the database. For example, you could bring the primary database into the `TRANSPORT-OFF` state to temporarily stop sending redo data to the standby databases. Then, to resume sending redo data to the standbys, you could bring the primary database into the `TRANSPORT-ON` state.



See Also:

[Managing the Members of a Broker Configuration](#) for more information about database state changes

Update Database Properties, As Needed

The broker enables you to set database properties, some of which correspond to database initialization parameters. You can change these properties to dynamically control such things

as redo transport, standby file management, log apply, and to support the overall configuration protection mode. The broker records the changes in the broker configuration file for each database in the Oracle Data Guard configuration and propagates the changes to the related initialization parameters in the server parameter files, if needed.

**See Also:**

[Managing the Members of a Broker Configuration and Oracle Data Guard Broker Properties](#) for complete information about database properties

Set Data Protection Modes, As Needed

The broker enables you to set the data protection mode for the configuration. You can configure the protection mode to maximize data protection, maximize availability, or maximize performance.

**See Also:**

[Managing Data Protection Modes](#) for information about managing data protection modes

Monitor the Configuration

You can check the health of the configuration, display and update the properties of the databases, and set Cloud Control events.

Cloud Control also provides a dynamic performance page that automatically and dynamically refreshes chart data and status at specified intervals. The performance chart shows a graphical summary of how far behind and how much redo data is being generated and applied.

Enable and Disable Operations

When you enable or disable a database in the broker configuration, you are effectively enabling or disabling the ability of the broker to manage and monitor the specified database.

The enable and disable operations are defined only for databases that are in a broker configuration; you cannot perform these broker operations on databases that are not part of the broker configuration.

However, disabling a broker configuration does not affect current services and operations in the actual Oracle Data Guard configuration. For example, when you disable a broker configuration, redo transport services and log apply services in the Oracle Data Guard configuration continue to function unchanged, but you can no longer manage them through the broker interfaces.

In addition, disabling a database *does not* remove or delete it from the broker configuration file. You can still change the properties of a disabled database and later reenabling your ability to manage with the broker using the DGMGRL `ENABLE`

CONFIGURATION or ENABLE DATABASE commands, or the Enable option in the Oracle Data Guard management pages of Cloud Control.

 **Note:**

If you disable broker management of a standby database in the broker configuration, that standby database cannot be used by the broker as a failover target in the event of loss of the primary database.

Disabling broker management of the configuration may be useful to do even though you are removing the broker's ability to monitor and control the databases. For example, it may be advantageous to disable a configuration temporarily in order to change one or more properties in the broker configuration all at the same time. When you change properties in a disabled configuration, it does not affect the actual database properties underneath because the changes are not applied to the running database until you reenables the configuration. For example, you might want to change the overall configuration protection mode and the redo transport services properties on a disabled configuration so that all changes are applied to the configuration at the same time upon the next enable operation.

 **See Also:**

[How the Protection Modes Influence Broker Operations](#)

Configuration Status

A configuration status reveals the overall health of the configuration.

Status of the configuration is acquired from the status of all of its databases.

The following list describes the possible status modes for a configuration:

- **Success**
The configuration, including all of the databases configured in it, is operating as specified by the user without any warnings or errors.
- **Warning**
One or more of the databases in the configuration are not operating as specified by the user. To obtain more information, use the DGMGRL `SHOW DATABASE <db-unique-name>` command or the Cloud Control display to locate each database and examine its status to reveal the source of the problem.
- **Error**
One or more of the databases in the configuration failed or may no longer be operating as specified by the user. To obtain more information, use the DGMGRL `SHOW DATABASE <db-unique-name>` command or the Cloud Control display to locate each database and examine its status to reveal the source of the problem.
- **Unknown/Disabled**

Broker management of the configuration is disabled and the broker is not monitoring the status of the databases in the configuration.

- **ROLLING DATABASE MAINTENANCE IN PROGRESS**

An operation performed using the PL/SQL `DBMS_ROLLING` package is in progress.

4

Managing the Members of a Broker Configuration

Learn how to manage broker configuration members, and monitor the state of each member of the configuration.

- [Managing Broker Configuration Members](#)
- [Managing States of Broker Configuration Members](#)
- [Managing Database Properties](#)
- [Managing Redo Transport Services](#)
- [Managing Redo Transport Services for Recovery Appliance](#)
- [Managing Log Apply Services](#)
- [Managing Data Protection Modes](#)
- [Managing Far Sync Instances](#)
- [Managing Fast-Start Failover](#)
- [Managing Database Conversions](#)
- [Database Status](#)

Managing Broker Configuration Members

The broker uses information in its configuration file to manage and monitor the state of each member of the configuration.

The broker distinguishes between different types of members in a broker configuration. Each type of member has a set of states and properties that are appropriate for that type of member.

Managing States of Broker Configuration Members

The members of a configuration can be in various states which, in an enabled configuration, determine the behavior of Oracle Data Guard.

The following table describes the various states.

Note that the following are not listed in the table:

- Snapshot standby databases (because they do not have states; they only receive redo data)
- Far sync instances (because they do not have states; they only receive redo and forward it to a standby database)
- Zero Data Loss Recovery Appliances (because they do not have states)

Table 4-1 Database States and Descriptions

Database Role	State Name	Description
Primary	TRANSPORT-ON	<p>Redo transport services are set up to transmit redo data to the standby databases or far sync instances when the primary database is open for read/write access.</p> <p>If this is an Oracle RAC database, all instances open in read/write mode will have redo transport services running.</p> <p>This is the default state for a primary database when it is enabled for the first time.</p>
Primary	TRANSPORT-OFF	<p>Redo transport services are stopped on the primary database.</p> <p>If this is an Oracle RAC database, redo transport services are not running on any instances.</p>
Physical standby	APPLY-ON	<p>Redo Apply is started on a physical standby database.</p> <p>If the standby database is an Oracle RAC database, the broker starts Redo Apply on exactly one standby instance, called the apply instance. If this instance fails, the broker automatically chooses another instance that is either mounted or open read-only. This new instance then becomes the apply instance.</p> <p>As of Oracle Database 12c Release 2 (12.2.0.1), Redo Apply can be set up to run on each active running physical standby instance. If a database has already been set up to run Redo Apply on multiple instances, then you can use the Data Guard broker property <code>ApplyInstances</code> to restrict the number of instances that are involved in Redo Apply on an Oracle RAC physical standby database.</p> <p>APPLY-ON is the default state for a physical standby database when it is enabled for the first time.</p> <p>If a license for the Oracle Active Data Guard option has been purchased, a physical standby database can be open while Redo Apply is active. This capability is known as real-time query. See <i>Oracle Data Guard Concepts and Administration</i> for more details.</p>
Physical standby	APPLY-OFF	<p>Redo Apply is stopped.</p> <p>If this is an Oracle RAC database, there is no instance running Apply Services until you change the database state to APPLY-ON.</p>

Table 4-1 (Cont.) Database States and Descriptions

Database Role	State Name	Description
Logical standby	APPLY-ON	<p>SQL Apply is started on the logical standby database when it is opened and the logical standby database guard is on.</p> <p>If this is an Oracle RAC database, SQL Apply is running on one instance, the apply instance. If this instance fails, the broker automatically chooses another open instance. This new instance becomes the apply instance.</p> <p>This is the default state for a logical standby database when it is enabled for the first time.</p>
Logical standby	APPLY-OFF	<p>SQL Apply is stopped. The logical standby database guard is on.</p> <p>If this is an Oracle RAC database, there is no instance running SQL Apply until you change the state to APPLY-ON.</p>

Database State Transitions

You can use the `DGMGRL EDIT DATABASE` command to explicitly change the state of a database.

For example, the `EDIT DATABASE` command in the following example changes the state of the `North_Sales` database to `TRANSPORT-OFF`.

```
DGMGRL> EDIT DATABASE 'North_Sales' SET STATE='TRANSPORT-OFF';
Succeeded.
```

The following sections describe in more detail the possible state transitions for primary and standby databases.

Primary Database State Transitions

When transitioning the primary database to the `TRANSPORT-ON` state, the broker sets up redo transport services using the redo transport-related properties of the configuration members and the `RedoRoutes` property at the primary database. Redo transport services setup is done by setting the `LOG_ARCHIVE_DEST_n` and `LOG_ARCHIVE_DEST_STATE_n` initialization parameters on the primary database, and the `LOG_ARCHIVE_CONFIG` initialization parameter on all databases (primary or standby) and far sync instances. If necessary, the broker also sets up the data protection mode of the database to match the protection mode recorded in the broker configuration file. Finally, if the database is open, the broker switches a log for each thread to initiate redo transport services.

When transitioning the primary database to the `TRANSPORT-OFF` state, the broker turns off redo transport services to all broker-managed standbys by resetting the `LOG_ARCHIVE_DEST_STATE_n` initialization parameter. Transmission of redo data to all broker-managed standbys is stopped. Log files continue to be archived at the primary database.

If the primary database is an Oracle RAC database, the broker configures redo transport services on all primary instances with the exact same settings.

Physical Standby Database State Transitions

When transitioning a physical standby database to the `APPLY-ON` state, the broker starts Redo Apply with options specified by the Redo Apply-related properties. If the standby is an Oracle RAC database, then the broker starts Redo Apply on one standby instance, called the apply instance.

Redo Apply can be set up to run on multiple active running physical standby instances. (This feature requires the standby database to have a license for the Oracle Active Data Guard option.) If a database has already been set up to run Redo Apply on multiple instances, then you can use the Data Guard broker property `ApplyInstances` to restrict the number of instances that are involved in Redo Apply on an Oracle RAC physical standby database.

If a license for the Oracle Active Data Guard option has been purchased, a physical standby database can be open while Redo Apply is active. This capability is known as real-time query. See *Oracle Data Guard Concepts and Administration* for more details.

The apply instance must be open before starting Redo Apply if any other instance is open.

When transitioning to the `APPLY-OFF` state, the broker stops Redo Apply.

Note:

If you perform online database relocation with Oracle RAC One Node on a physical standby, then the new instance is started in the same mode as the currently running instance. Therefore, if the database is mounted on the original instance, then the database will be mounted on the new instance. Likewise, if the database is open on the original instance, then the database will be open on the new instance. This may result in the new instance starting in a mode that does not match the start option recorded with Oracle Clusterware for the database.

Logical Standby Database State Transitions

When transitioning a logical standby database to the `APPLY-ON` state, the broker will wait until the database is open, and then enable the database guard to prevent modifications to tables in the logical standby database, and start SQL Apply with options specified by the log apply-related properties. If the logical standby database is an Oracle RAC database, the broker starts SQL Apply on one standby instance, the apply instance.

When transitioning to the `APPLY-OFF` state, the broker stops SQL Apply.

 **See Also:**

- [Managing Log Apply Services](#) for information about managing SQL Apply
- [Managing Redo Transport Services](#) for more details on managing redo transport services and a list of redo transport-related properties
- *Oracle Data Guard Concepts and Administration* for information about the database guard
- [Oracle Data Guard Command-Line Interface Reference](#) for complete information about the `EDIT DATABASE` command.

Managing Database Properties

There are two types of database properties: monitorable and configurable.

Both monitorable and configurable properties can be further divided into properties having database-wide scope or instance-specific scope.

- Monitorable property values can be viewed only when the associated database is enabled.

Monitorable properties allow you to view run-time information related to databases, but you cannot change the values of these properties.

- Configurable property values can be viewed and dynamically updated.

Configurable properties affect the operation or configuration of the broker. You can change the value of these properties using DGMGRL or Cloud Control. You can edit properties whether the configuration and its members are enabled or disabled. However, if a configuration member is disabled, then the new property value does not take effect until you enable the configuration or member, as appropriate.

 **See Also:**

[Oracle Data Guard Broker Properties](#) for a detailed list of all database properties

To see these properties, you can use the DGMGRL `SHOW` command or Edit Properties page in Cloud Control. [Example 4-1](#) uses the `SHOW DATABASE VERBOSE` command to display information about the `North_Sales` database.

 **See Also:**

[Oracle Data Guard Command-Line Interface Reference](#) for complete information about the DGMGRL command-line interface

Example 4-1 Using the SHOW DATABASE VERBOSE Command to Display Properties

```
SHOW DATABASE VERBOSE 'North_Sales';
```

```
Database - North_Sales
```

```
Role:                PRIMARY
Intended State:      TRANSPORT-ON
Instance(s):
  North_sales1
```

```
Properties:
```

```
DGConnectIdentifier      = 'North_Sales.example.com'
ObserverConnectIdentifier = ''
FastStartFailoverTarget  = ''
PreferredObserverHosts   = ''
LogShipping              = 'ON'
RedoRoutes                = ''
LogXptMode               = 'ASYNC'
DelayMins                 = '0'
Binding                  = 'optional'
MaxFailure                = '0'
ReopenSecs               = '30'
NetTimeout               = '300'
RedoCompression          = 'DISABLE'
PreferredApplyInstance    = ''
ApplyInstanceTimeout      = '0'
ApplyLagThreshold         = '0'
TransportLagThreshold     = '0'
TransportDisconnectedThreshold = '0'
ApplyParallel            = 'AUTO'
ApplyInstances            = '0'
ArchiveLocation           = ''
AlternateLocation         = ''
StandbyArchiveLocation   = ''
StandbyAlternateLocation  = ''
InconsistentProperties    = '(monitor)'
InconsistentLogXptProps  = '(monitor)'
LogXptStatus              = '(monitor)'
SendQEntries              = '(monitor)'
RecvQEntries              = '(monitor)'
HostName                  = 'North_Sales.example.com'
StaticConnectIdentifier    = ''
'(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)
(HOST=North_Sales.example.com) (PORT=2840))
(CONNECT_DATA=(SERVICE_NAME=North_Sales_DGMGRL.example.com)
(INSTANCE_NAME=north_sales1) (SERVER=DEDICATED)))'
```

```
Log file locations:
Alert log                : /dev/oracle/log/diag/rdbms/North_Sales/
north1/trace/alert_north1.log
Data Guard Broker log    : /dev/oracle/log/diag/rdbms/North_Sales/
```

```
north1/trace/drcnorth1.log
```

```
Database Status:  
SUCCESS
```

Monitorable (Read-Only) Properties

Monitorable properties allow you to view information related to a configuration member, but you cannot change the property values.

Monitorable properties can be very helpful when you are trying to diagnose problems in the broker configuration. For example, you can view the `InconsistentLogXptProps` monitorable property to determine where there is a discrepancy in redo transport services properties between the broker configuration file and the actual value currently used by the database.

You can list all monitorable properties using the `DGMGRL SHOW DATABASE VERBOSE` command. Use the `SHOW DATABASE` command to obtain more details about a particular property. For example, the following shows the `InconsistentLogXptProps` property:

```
DGMGRL> SHOW DATABASE 'North_Sales' 'InconsistentLogXptProps';
```

```
INCONSISTENT LOG TRANSPORT PROPERTIES
```

INSTANCE_NAME	STANDBY_NAME	PROPERTY_NAME	MEMORY_VALUE	BROKER_VALUE
south_sales1	South_Sales	ReopenSecs	600	300

Cloud Control displays the information obtained from these properties on the Edit Properties page.

Configurable (Changeable) Properties

Configurable properties affect the operation or configuration of a database or far sync instance.

When you use `DGMGRL` or Cloud Control to create a primary database and import existing standby databases and far sync instances into a new broker configuration, the property values are initially imported from the database or far sync instance settings.

You can update many property values when a configuration member is either disabled or enabled. When a new member is added into the configuration, the broker connects to that member and imports initial values for the member's properties from the current member settings. For example:

```
DGMGRL> SHOW DATABASE 'North_Sales' LogXptMode;  
LogXptMode = ASYNC
```

```
DGMGRL> EDIT DATABASE 'North_Sales' SET PROPERTY LogXptMode='SYNC';  
Property "LogXptMode" updated
```

```
DGMGRL> SHOW DATABASE 'North_Sales' LogXptMode;  
LogXptMode = SYNC
```

When the configuration is enabled, for properties related to redo transport, the broker keeps the member property values in the broker configuration file consistent with the values being

used by the member. The broker no longer maintains values for properties related to initialization parameters in the configuration file. Note that there is no issue with maintaining consistency. Although the broker no longer maintains values for these properties, it is still possible to use the broker CLI to update and examine the value of these properties. Due to user action, it is possible for the parameter value in the system global area (SGA) to differ from the parameter value in the server parameter file. This is not flagged as an inconsistency and only means that the server parameter file value will take effect the next time the database is restarted. The specified database must be accessible to be able to make these property changes.

**Note:**

Even though you can change a property value when the configuration is disabled, the change does not take effect on the configuration member unless the configuration is enabled. Also note that some property values can only be changed in the disabled state.

Resetting Broker Configurable Properties to Default Values

Most broker configurable properties have a default value, but you can specify a different value to override it.

The broker recognizes when a default value has been restored for a property and no longer considers it a user-supplied value. This is expedient in upgrade scenarios because if a default value for a property changes between releases, then the new default value is automatically put into effect after an upgrade is complete. Values that are considered user-supplied are not automatically upgraded.

You do not need to know the actual default value in order to reset it. You can reset a default value at the configuration, configuration member, or instance level.

**See Also:**

- ["EDIT CONFIGURATION RESET \(Property\)"](#)
- ["EDIT DATABASE RESET \(Property\)"](#)
- ["EDIT FAR_SYNC RESET \(Property\)"](#)
- [EDIT RECOVERY_APPLIANCE RESET \(Property\)](#)

Managing Redo Transport Services

To manage redo transport services, you specify configurable properties on each configuration member.

The properties you specify are as follows::

- `DGConnectIdentifier`
- `StandbyAlternateLocation`

- Binding
- Encryption (This property can be set only on a Recovery Appliance)
- LogShipping
- LogXptMode
- MaxFailure
- NetTimeout
- RedoCompression
- RedoRoutes
- ReopenSecs
- StandbyArchiveLocation

You can use these properties to specify how the broker configures redo transport services for the standby database. The actual redo transport setup, such as setting the `LOG_ARCHIVE_DEST_n` initialization parameter, is carried out by the broker on the primary database (except for the `StandbyArchiveLocation` property). If changing the property requires that you change the `LOG_ARCHIVE_DEST_n` initialization parameter attributes, the broker forces a log switch on each thread so that the new setting is adopted immediately by the primary database.

You should also preset these properties on the primary database in preparation for it to be switched over to a standby database.

Setting Up For Redo Transport

Redo data is transported to a standby database using Oracle Net.

An Oracle Net service name is specified with the `SERVICE` attribute of the `LOG_ARCHIVE_DEST_n` initialization parameter and is used to transmit redo data to the standby database. The Oracle Net service name is translated into a connect descriptor that contains the information necessary for connecting to the standby database.

The `SERVICE` attribute can be set or changed by using the `DGConnectIdentifier` database property. The `DGConnectIdentifier` property is set when a database is first added to the configuration. Its initial value is the connect identifier that is specified in the optional `CONNECT IDENTIFIER IS` clause of the `ADD DATABASE` command.

The `DGConnectIdentifier` property value is also used to set up the `FAL_SERVER` initialization parameter. If the `DGConnectIdentifier` property for any database is changed, the `SERVICE` attribute of the corresponding `LOG_ARCHIVE_DEST_n` initialization parameter will also be changed. In addition, the `FAL_SERVER` initialization parameter will also be updated on every enabled standby database in the configuration.

Managing Redo Transport Services for Data Protection Modes

As a part of the overall configuration protection mode, you must ensure that redo transport services are also properly set up for the data protection mode that you choose.

[Managing Data Protection Modes](#) describes how the broker handles data protection modes.

You use the `LogXptMode` or `RedoRoutes` database properties to set the `SYNC`, `ASYNC`, or `FASTSYNC` mode for redo transport services.

The following redo transport modes are supported:

SYNC

Configures redo transport services for this standby database using the `SYNC` and `AFFIRM` attributes of the `LOG_ARCHIVE_DEST_n` initialization parameter. This mode, along with standby redo log files, is required for configurations operating in either maximum protection mode or maximum availability mode. This redo transport service enables the highest grade of data protection to the primary database, but also can incur a higher performance impact.

ASync

Configures redo transport services for this standby database using the `ASync` and `NOAFFIRM` attributes of the `LOG_ARCHIVE_DEST_n` initialization parameter. This mode, along with standby redo log files, enables a moderate grade of protection to the primary database, and lower performance impact.

FASTSync

Configures redo transport services for this standby database using the `SYNC` and `NOAFFIRM` attributes of the `LOG_ARCHIVE_DEST_n` initialization parameter. This mode is only available in maximum availability protection mode.

Advanced Redo Transport Settings

You can use the `RedoRoutes` property to override the default behavior in which a primary database sends its redo to every possible redo transport destination in the configuration.

An example of redo transport topology other than the default would be one in which a physical standby or a far sync instance forwards redo received from the primary database to one or more destinations, or one in which the redo transport mode used for a given destination is dependent on which database is in the primary role.



See Also:

- [RedoRoutes](#) for information about redo routing rules when you use the `RedoRoutes` property

Example 4-2 Using the RedoRoutes Property for Real-Time Cascading

Consider a configuration that has a primary database (`North_Sales`) and two physical standby databases (`Local_Sales` and `Remote_Sales`). The `Local_Sales` database is located in the same data center as the primary for high availability purposes. The `Remote_Sales` database is located in a remote data center for disaster recovery purposes. Instead of the primary having to ship its redo to both databases, it is possible to use the `RedoRoutes` property to configure real-time cascading, in which the local physical standby database forwards redo from `North_Sales` to the remote physical standby database, `Remote_Sales`. To accomplish this, the `RedoRoutes` property must be set as follows:

- On the `North_Sales` database, the `RedoRoutes` property must specify that if `North_Sales` is in the primary role, then it should ship redo to the `Local_Sales` database using synchronous transport mode. This rule prevents the primary from shipping redo data directly to the `Remote_Sales` database.
- On the `Local_Sales` database, the `RedoRoutes` property must specify that if `North_Sales` is in the primary role, then `Local_Sales` should forward redo it receives from `North_Sales` on to `Remote_Sales`.

```
DGMGRL> EDIT DATABASE 'North_Sales' SET PROPERTY 'RedoRoutes' = '(LOCAL : Local_Sales SYNC)';
DGMGRL> EDIT DATABASE 'Local_Sales' SET PROPERTY 'RedoRoutes' = '(North_Sales : Remote_Sales ASYNC)';
```

To see the runtime `RedoRoutes` configuration, use the `SHOW CONFIGURATION` command. For example:

```
DGMGRL> SHOW CONFIGURATION;
```

```
Configuration - Sales_Configuration
```

```
Protection Mode: MaxAvailability
Members:
North_Sales - Primary database
Local_Sales - Physical standby database
Remote_Sales - Physical standby database (receiving current redo)
```

```
Fast-Start Failover: DISABLED
```

```
Configuration Status:
SUCCESS
```

Note that the `ASYNC` redo transport attribute was explicitly specified in the redo route rule for the `Remote_Sales` destination to enable real-time cascading of redo to that destination. (Real-time cascading requires a license for the Oracle Active Data Guard option.)

To disable real-time cascading of redo, do not specify the `ASYNC` redo transport attribute. For example:

```
DGMGRL> EDIT DATABASE 'Local_Sales' SET PROPERTY 'RedoRoutes' = '(North_Sales : Remote_Sales)';
```

Example 4-3 Using the `RedoRoutes` Property for Remote Alternate Destinations

The `RedoRoutes` property can also be used to set up a remote alternate destination so that a terminal member can still receive redo data even if the member from which it was receiving the redo data fails. Using the previous example, it would be possible to have the primary database, `North_Sales`, send redo data directly to `Remote_Sales` if the `Local_Sales` standby database failed. It is also possible, using the `PRIORITY` attribute, to set it up so that once the `Local_Sales` failure has been resolved it can resume shipping redo to `Remote_Sales`.

```
DGMGRL> EDIT DATABASE 'North_Sales' SET PROPERTY 'RedoRoutes' = '(LOCAL : ( Local_Sales ASYNC PRIORITY=1, Remote_Sales ASYNC PRIORITY=2 ))';
Property "RedoRoutes" updated
```

```
DGMGRL> EDIT DATABASE 'Local_Sales' SET PROPERTY 'RedoRoutes' = '(North_Sales : Remote_Sales ASYNC)';
Property "RedoRoutes" updated
```

```
DGMGRL> SHOW CONFIGURATION;
```

```
Configuration - Sales_Configuration
```

```
Protection Mode: MaxPerformance
```

```
Members:
```

```
North_Sales - Primary database
```

```
Local_Sales - Physical standby database
```

```
Remote_Sales - Physical standby database
```

```
Fast-Start Failover: DISABLED
```

```
Configuration Status:
```

```
SUCCESS
```

To see the full RedoRoutes configuration, use the SHOW CONFIGURATION VERBOSE command. For example:

```
DGMGRL> SHOW CONFIGURATION VERBOSE;
```

```
Configuration - Sales_Configuration
```

```
Protection Mode: MaxPerformance
```

```
Members:
```

```
North_Sales - Primary database
```

```
Local_Sales - Physical standby database
```

```
Remote_Sales - Physical standby database
```

```
Remote_Sales - Physical standby database (alternate of  
Local_Sales)
```

```
Properties:
```

```
FastStartFailoverThreshold = '180'
```

```
OperationTimeout = '30'
```

```
TraceLevel = 'USER'
```

```
FastStartFailoverLagLimit = '300'
```

```
CommunicationTimeout = '180'
```

```
ObserverReconnect = '0'
```

```
FastStartFailoverAutoReinstate = 'TRUE'
```

```
FastStartFailoverPmyShutdown = 'TRUE'
```

```
BystandersFollowRoleChange = 'ALL'
```

```
ObserverOverride = 'FALSE'
```

```
ExternalDestination1 = ''
```

```
ExternalDestination2 = ''
```

```
PrimaryLostWriteAction = 'CONTINUE'
```

```
ConfigurationWideServiceName = 'c0_CFG'
```

```
Fast-Start Failover: DISABLED
```

```
Configuration Status:
```

```
SUCCESS
```

Turning Redo Transport Services On and Off

You turn redo transport services on and off by setting the state of the primary database.

Setting the primary database state to `TRANSPORT-ON` starts redo transport services at the primary database, and setting the primary database state to `TRANSPORT-OFF` stops redo transport services at the primary database.

 **Note:**

Oracle does not recommend turning off redo transport services to all standby databases. This increases the risk of data loss if the primary database fails.

Turn redo transport services on and off to an individual standby database using the `LogShipping` database property on the standby database. The `LogShipping` property accepts values `ON` and `OFF`. If you set the `LogShipping` property to `OFF` for a standby database, redo transport services to this standby database are turned off, while redo transport services to other databases are not affected. You can set `LogShipping` to `ON` to turn back on redo transport services to the standby database.

The relationship between setting the primary database state and setting the `LogShipping` property is as follows:

- If the primary database state is set to `TRANSPORT-OFF`, redo transport services to all the standby databases are stopped regardless of the `LogShipping` property values of the individual standby databases.
- If the primary database state is set to `TRANSPORT-ON`, redo transport services to each standby database are determined by the `LogShipping` property of that database.

[Example 4-4](#) and [Example 4-5](#) show how to turn off redo transport services in two different scenarios.

Example 4-4 Turn Off Redo Transport Services to All Standby Databases

```
DGMGRL> EDIT DATABASE 'North_Sales' SET STATE='TRANSPORT-OFF';
DGMGRL> SHOW DATABASE 'North_Sales';
```

```
Database - North_Sales

Role:                PRIMARY
Intended State:      TRANSPORT-OFF
Instance(s):
  north_sales1

Database Status:
SUCCESS
```

Example 4-5 Turn Off Redo Transport Services to a Specific Standby Database

```
DGMGRL> EDIT DATABASE 'South_Sales' SET PROPERTY 'LogShipping'='OFF';
Property "LogShipping" updated

DGMGRL> SHOW DATABASE 'South_Sales' 'LogShipping';
LogShipping = 'OFF'
```

Specifying Locations for Archived Redo Log Files

You can use broker configurable properties to set up locations to store both archived standby redo log files and archived online redo log files.

For online redo log files, use the `ArchiveLocation` and `AlternateLocation` properties on a primary, logical, or snapshot standby database.

For standby redo log files, use the `StandbyArchiveLocation` and `StandbyAlternateLocation` properties on a standby database. If the `StandbyArchiveLocation` is not set, `ArchiveLocation` or `AlternateLocation` specify the archiving location for both of online and standby redo log files. If `StandbyArchiveLocation` is set, `ArchiveLocation` and `AlternateLocation` specify the archiving location for online redo log files.

The `StandbyArchiveLocation` property specifies a location to store archived redo log files. The broker uses the location to store only archived redo log files received from the primary database. For archived redo log files generated locally when the database is either the primary database, a logical standby database, or a snapshot standby database, you need to specify the `ArchiveLocation` property. The broker allows the value of the `StandbyArchiveLocation` and `ArchiveLocation` properties to be the same as the location you set up for locally generated logs, in which case the broker sets up the `VALID_FOR` attribute of the destination appropriately so that it can be used for both the archived redo log files received from the primary database and archived redo log files generated locally. When a location is used for both online and standby redo log files, you must configure the `ArchiveLocation` property to the shared location and leave the `StandbyArchiveLocation` property empty.

You can also set up an alternate location to store archived redo log files by using the `StandbyAlternateLocation` and `AlternateLocation` properties. The alternate locations specified by these properties are where the archived redo log files are stored if the original archive location (specified by `StandbyArchiveLocation` or `ArchiveLocation`) fails. The broker sets up the alternate location properly using the `ALTERNATE` attribute of the `LOG_ARCHIVE_DEST_n` initialization parameter.

 **Note:**

You can use the database recovery area to store archived redo log files on the standby. In such a case, the value of the `StandbyArchiveLocation` or `ArchiveLocation` properties can be set to `USE_DB_RECOVERY_FILE_DEST`.

If you are not using a database recovery area, then on a logical standby database Oracle recommends that the `ArchiveLocation` property be different from the value of the `StandbyArchiveLocation` property.

Example 4-6 Using the Same Archiving Location for Standby Redo Log Files and Online Redo Log Files

The following example uses the same archiving location for online redo log files and standby redo log files.

```
ArchiveLocation='/archfs/arch'
```

Example 4-7 Specifying an Alternate Location for Archived Standby and Online Redo Log Files

The following example uses the same archiving location for online redo log files and standby redo log files. Configuring the `AlternateLocation` property ensures that a shared, alternate, location is available to store standby redo log files if the location specified by the `ArchiveLocation` property is unavailable.

```
ArchiveLocation='/archfs/arch'  
AlternateLocation='/archfs/alt'
```

Example 4-8 Specifying Separate Archiving Locations for Online and Standby Redo Log Files

This example configures separate archiving locations for standby redo log files and online redo log files. Also, alternate archiving locations are configured for the online and standby redo log files by using the `AlternateLocation` and `StandbyAlternateLocation` properties respectively

```
ArchiveLocation='/archfs/arch/online'  
AlternateLocation='/archfs/alt/online'  
StandbyArchiveLocation='/archfs/arch/standby'  
StandbyAlternateLocation='/archfs/alt/standby'
```

Other Redo Transport Settings

You can use database properties to tune the performance of redo transport services and to set up redo transport services failure policies.

The properties used are: `Binding`, `MaxFailure`, `NetTimeout`, `RedoCompression`, and `ReopenSecs`. These properties correspond to attributes on the `LOG_ARCHIVE_DEST_n` initialization parameter.

**See Also:**

[Oracle Data Guard Broker Properties](#) for complete information about these database properties

Redo Transport Services in an Oracle RAC Database Environment

If the primary database is an Oracle RAC database, the broker ensures that redo transport services are set up identically on each of the primary database instances.

Each instance has the same remote destinations, and for each remote destination, all instances are set up the same in terms of redo transport service, performance related settings, local archival of the online redo logs, and so on. If an instance has different settings, the broker raises a health check warning on that particular instance

Settings relative to redo transport services are saved in the broker configuration file as properties. When you update a redo transport-related property on a standby database, the corresponding change is also made automatically by the broker to the `LOG_ARCHIVE_DEST_n` initialization parameter on all of the primary database instances. If a new instance comes up on the primary database, the broker sets up redo transport services for the new instance using the redo transport-related properties of all the configuration members currently being managed by the broker. After the new instance is opened for activity, all archived redo log files generated on this instance begin to transmit to members of the configuration.



See also:

Oracle Data Guard Concepts and Administration for additional information about the `LOG_ARCHIVE_DEST_n` initialization parameter

Transport Lag

Transport lag is a measure of the degree to which the transport of redo to a standby database or far sync instance lags behind the generation of redo on the primary database.

If there are one or more redo gaps on a standby database or far sync instance, then the transport lag is calculated as if no redo has been received after the beginning of the earliest redo gap.

Both Cloud Control and the DGMGRL client display the redo transport lag for each managed standby or far sync instance. Cloud Control displays the transport lag on the Oracle Data Guard home page. The DGMGRL client displays the transport lag in the `SHOW DATABASE` output. There is no transport lag displayed for a primary database. Here is an example of the transport lag for a physical standby database:

```
DGMGRL> show database 'South_Sales';
```

```
Database - South_Sales
```

```
Role:                PHYSICAL STANDBY
Intended State:      APPLY-ON
Transport Lag:    0 seconds (computed 0 seconds ago)
Apply Lag:           0 seconds (computed 0 seconds ago)
Apply Rate:          255.00 KByte/s
Real Time Query:    OFF
Instance(s):         south_sales1
```

```
Database Status:  
SUCCESS
```

Starting with Oracle Database Release 19c, the `SHOW CONFIGURATION LAG` command displays a summary of the broker configuration and the transport lag of all standby databases.

The transport lag can help you identify any problems that may exist with the redo transport services.

You can set the `TransportLagThreshold` database configurable property to generate a health check warning when the transport of redo data to a standby database or far sync instance lags behind the generation of redo data on the primary database.

The following command sets the `TransportLagThreshold` property to 15 seconds:

```
DGMGRL> EDIT DATABASE 'South_Sales' SET PROPERTY 'TransportLagThreshold'=15;  
Property TransportLagThreshold updated
```

Additionally, you can set the `TransportDisconnectedThreshold` database configurable property to generate a health check warning if a standby or far sync instance finds that it has not had any redo transport-related communication with the primary database. The property has a default value of 30 seconds.

The following command sets the `TransportDisconnectedThreshold` property to 15 seconds:

```
DGMGRL> EDIT DATABASE 'South_Sales' SET PROPERTY 'TransportDisconnectedThreshold'=15;  
Property TransportDisconnectedThreshold updated
```

Managing Redo Transport Services for Recovery Appliance

Redo transport services for a Zero Data Loss Recovery Appliance (Recovery Appliance) are managed in the same way as for a standby database.

You can use the broker to manage redo transport services to a Recovery Appliance from any member in the configuration. For example, to add a Recovery Appliance to a broker configuration and then enable it, you would take the following steps:

1. Add the Recovery Appliance to the broker configuration.

```
DGMGRL> ADD RECOVERY_APPLIANCE EnterpriseRecoveryAppliance AS CONNECT IDENTIFIER IS  
EnterpriseRecoveryAppliance.example.com;  
Oracle Recovery Appliance "EnterpriseRecoveryAppliance" added
```

```
DGMGRL> SHOW RECOVERY_APPLIANCE 'EnterpriseRecoveryAppliance';  
Oracle Recovery Appliance - EnterpriseRecoveryAppliance  
Transport Lag: 0 seconds  
Redo Source: North_Sales
```

```
Oracle Recovery Appliance Status:  
DISABLED
```

2. Enable the Recovery Appliance.

```
DGMGRL> ENABLE RECOVERY_APPLIANCE 'EnterpriseRecoveryAppliance';
```

```
DGMGRL> SHOW RECOVERY_APPLIANCE 'EnterpriseRecoveryAppliance';  
Oracle Recovery Appliance - EnterpriseRecoveryAppliance  
Transport Lag: 0 seconds  
Redo Source: North_Sales
```

```
Oracle Recovery Appliance Status:
SUCCESS
```

3. Set a transport lag threshold.

Set the `TransportLagThreshold` database configurable property to generate a health check warning when the transport of redo data to the Recovery Appliance lags behind the generation of redo data on the source database. This step is optional, but if you do not specify a transport lag threshold, then the default value of 30 seconds is used and a warning is generated if a transport lag exceeds 30 seconds.

The following command sets the `TransportLagThreshold` property to 15 seconds:

```
DGMGR> EDIT RECOVERY_APPLIANCE 'EnterpriseRecoveryAppliance'
SET PROPERTY 'TransportLagThreshold'=15;
Property TransportLagThreshold updated
```

Example 4-9 Setting Up Redo Transport From a Physical Standby To a Recovery Appliance

Consider a configuration that has a primary database (`North_Sales`) a physical standby database (`South_Sales`), and a Recovery Appliance (`EnterpriseRecoveryAppliance`). Instead of the primary having to ship its redo to both the standby and the Recovery Appliance, you can set up the `RedoRoutes` property so that the primary sends redo only to the physical standby, and the physical standby then forwards that redo to the Recovery Appliance. To accomplish this, the `RedoRoutes` property must be set as follows:

- On the `North_Sales` database, the `RedoRoutes` property must specify that if `North_Sales` is in the primary role, then it should ship redo to the `South_Sales` database using synchronous transport mode. This rule prevents the primary from shipping redo data directly to the Recovery Appliance, `EnterpriseRecoveryAppliance`.
- On the `South_Sales` database, the `RedoRoutes` property must specify that if `North_Sales` is in the primary role, then `South_Sales` should forward redo it receives from `North_Sales` on to `EnterpriseRecoveryAppliance`.

```
DGMGR> EDIT DATABASE 'North_Sales' SET PROPERTY 'RedoRoutes' = '(LOCAL :
South_Sales SYNC)';
DGMGR> EDIT DATABASE 'South_Sales' SET PROPERTY 'RedoRoutes' = '(North_Sales :
EnterpriseRecoveryAppliance ASYNC)';
```

Note that the `ASYNC` redo transport attribute was explicitly specified in the redo route rule for the `EnterpriseRecoveryAppliance` destination to enable real-time cascading of redo to that destination.

Once a Recovery Appliance has been added to a broker configuration, it can receive redo from either the primary database or a standby database. [Example 4-9](#) shows how to set up a Recovery Appliance to receive redo from a physical standby database.

See Also:

- [Managing Redo Transport Services](#) for information about managing redo transport services

Managing Log Apply Services

You can manage Redo Apply and SQL Apply on physical and logical standby databases by using database properties related to log apply.

The database properties related to log apply are as follows:

- Properties common to Redo Apply and SQL Apply
 - `ApplyInstanceTimeout`
 - `DelayMins`
 - `PreferredApplyInstance`
- Properties specific to Redo Apply
 - `ApplyParallel`
 - `ApplyInstances`

There are some properties related to SQL Apply that, if changed, may require a restart of SQL Apply if the current database state is `APPLY-ON`. See the information in [Oracle Data Guard Broker Properties](#) about properties related to SQL Apply, to determine which ones require SQL Apply to be restarted.

If the current database state is `APPLY-OFF`, the property changes will take effect the next time the database state is changed to `APPLY-ON`.

Managing Delayed Apply

You can set up Apply Services so that the application of redo to the standby database is delayed.

This allows the standby database to lag behind the primary database, and if a user error (for example, dropping a table) occurs during this window of time, the standby database will still contain the correct data that can be transmitted back to the primary database to repair the data.

By default, no delay is configured and the redo data is applied on a standby database as soon as possible. If the standby database has standby redo logs configured, the broker will enable real-time apply. When Redo Apply and SQL Apply apply redo in real time, the redo data is recovered directly from the standby redo log files as they are being filled. This means that the standby database does not have to wait for the log files to be archived before applying redo data from the archived redo log files. This minimizes the transactional lag between the primary and the standby.

Use the `DelayMins` database property to specify the number of minutes that log apply services must wait before applying redo data to the standby database. Note that only log apply services on the standby database are delayed. Redo transport services on the primary database are not delayed, thus the primary database data is still well protected by the standby database.

 **Caution:**

Because the broker automatically enables real-time apply on standby databases, Oracle recommends that you configure all databases to use Flashback Database.

Managing Parallel Apply with Redo Apply

For Redo Apply, you can configure whether multiple parallel processes are used to apply redo data received from the primary database by using the `ApplyParallel` database property.

Parallelism is enabled by default, which means Redo Apply automatically chooses the optimal number of parallel processes based on the number of CPUs in the system. (This is equivalent to setting the `ApplyParallel` property to `AUTO`.) You can disable parallelism by setting the `ApplyParallel` property to `NO`.

 **Note:**

Parallel Redo Apply is different from multi-instance Redo Apply. Parallel Redo Apply means that there are multiple Redo Apply slaves per instance; this value is set using the broker `ApplyParallel` property. Multi-instance Redo Apply means that there are multiple instances running Redo Apply; this value is set with the broker `ApplyInstances` property. The two properties can be used together to control the Redo Apply slaves on each instance on which apply is running in multi-instance apply. The number of parallel slaves specified by the `ApplyParallel` property will be the same on each instance in a multi-instance apply configuration.

 **See Also:**

[ApplyParallel](#)

[ApplyInstances](#)

Managing Multi-Instance Redo Apply

For Redo Apply in an Oracle RAC database, you can configure the number of instances that can be engaged in recovery by means of the `ApplyInstances` property value.

By default, only one instance is involved in recovery activity. However, the `ApplyInstances` property can be set to indicate a specific number of instances or the value `ALL`, to indicate all instances. When recovery is started, it checks to see if enough instances as configured are available to start recovery on. If not, then broker delays starting recovery for one minute to allow other instances to start up and then starts recovery.

During periodic health checks, broker checks to see if more instances have started that could potentially be engaged in recovery. If so, then broker stops and restarts recovery to engage the additional instances.

Changing the value of the `ApplyInstances` property value results in recovery being restarted with the new values.

All instances must be in the same state (open or mounted) to be able to engage that instance in recovery.

Apply Services in an Oracle RAC Database Environment

When a standby database is an Oracle RAC database, SQL Apply and Redo Apply make use of an apply instance.

SQL Apply can run on only one instance of an Oracle RAC database at any time. This instance is called the apply instance.

Redo Apply can run on all instances of an Oracle RAC database at the same time, but only one of the instances is the apply coordinator and that is the instance that the broker considers to be the apply instance. This feature requires that the `ApplyInstances` database configurable property (valid only on physical standby databases) be set to a non-zero value. See [ApplyInstances](#).

If the apply instance fails, then the broker automatically restarts SQL Apply, or the Redo Apply coordinator, as appropriate, on a different instance. This is called apply instance failover (see [Apply Instance Failover](#)).

Selecting the Apply Instance

If you have no preference which instance is to be the apply instance in an Oracle RAC standby database, the broker randomly picks an apply instance. If you want to select a particular instance as the apply instance, there are two methods to do so.

Note:

The information in this section is not applicable to snapshot standby databases or far sync instances.

- The first method is to set the value of the `PreferredApplyInstance` database property to the name of the instance (see the `InstanceName` property) you want to be the apply instance. The broker starts log apply services on that instance if no apply instance is yet selected in the Oracle RAC standby database. This could be the case before you enable the standby database for the first time, or if the apply instance just failed and the broker is about to do an apply instance failover, or if the Oracle RAC database is currently the primary and you want to specify its apply instance in preparation for a switchover. Once the apply instance is selected and, as long as the apply instance is still running, the broker disregards the value of the `PreferredApplyInstance` property even if you change it.
- The second method is to change the apply instance when the apply instance is already selected and is running. To change the apply instance, issue the `DGMGRL SET STATE` command to set the standby database state to `APPLY-ON`, with a specific apply instance

argument. The `SET STATE` command will update the `PreferredApplyInstance` property to the new apply instance value, and then move log apply services to the new instance. For example, use `DGMGRL SHOW` command to show the available instances for the standby database, then issue the `EDIT DATABASE` command to move log apply services to the new instance:

```
DGMGRL> SHOW DATABASE 'South_Sales'

Database - South_Sales

Role:                PHYSICAL STANDBY
Intended State:      APPLY-ON
Transport Lag:       0 seconds (computed 1 second ago)
Apply Lag:           0 seconds (computed 1 second ago)
Apply Rate:          1017.00 KByte/s
Real Time Query:    OFF
Instance(s):
  south_sales1      (apply instance)
  south_sales2

Database Status:
SUCCESS

DGMGRL> EDIT DATABASE 'South_Sales' SET STATE='APPLY-ON' WITH APPLY
INSTANCE='south_sales2';
Succeeded.

DGMGRL> SHOW DATABASE 'South_Sales' 'PreferredApplyInstance';
PreferredApplyInstance = 'south_sales2'

DGMGRL> SHOW DATABASE 'South_Sales'

Database - South_Sales

Role:                PHYSICAL STANDBY
Intended State:      APPLY-ON
Transport Lag:       0 seconds (computed 1 second ago)
Apply Lag:           0 seconds (computed 1 second ago)
Apply Rate:          1017.00 KByte/s
Real Time Query:    OFF
Instance(s):
  south_sales1
  south_sales2      (apply instance)

Database Status:
SUCCESS
```

Ensure that the new apply instance is running when the command is issued. Otherwise, the apply instance remains the same.

Once the apply instance is selected, the broker keeps apply instance information in the broker configuration file so that even if the standby database is shut down and restarted, the broker still selects the same instance to start log apply services. The apply instance remains unchanged until changed by the user or it fails for any reason and the broker decides to do an apply instance failover.

Apply Instance Failover

To tolerate a failure of the apply instance, the broker leverages the availability of the Oracle RAC standby database by automatically failing over log apply services to a different standby instance.

The apply instance failover capability provided by the broker enhances data protection.

To set up apply instance failover, set the `ApplyInstanceTimeout` database property to specify the time period that the broker will wait after detecting an apply instance failure and before initiating an apply instance failover. To select an appropriate timeout value, you need to consider:

- If there is another mechanism in the cluster (such as Oracle Clusterware) that will try to recover the failed apply instance.
- How long your business can tolerate not applying redo data on the standby database.
- The overhead associated with moving the log apply services to a different instance.

The broker default value of the `ApplyInstanceTimeout` property is 0 seconds, indicating that apply instance failover should occur immediately upon detection of the failure of the current apply instance.

After the broker initiates an apply instance failover, the broker selects a new apply instance according to the following rule: if the `PreferredApplyInstance` property indicates an instance that is currently running, select it as the new apply instance; otherwise pick a random instance that is currently running to be the new apply instance.

In addition, if the physical standby database was operating in real-time query mode when the apply instance failed, then after Oracle recovery cleanup is completed, the broker opens any instances that had been automatically closed. If the failed apply instance was the only instance open, then the instance chosen as the new apply instance is opened before starting apply services so that real-time query is once again in effect.

See Also:

- *Oracle Data Guard Concepts and Administration* for more information about real-time query mode
- The My Oracle Support note 1357597.1 at <http://support.oracle.com> for additional information about apply instance failures in an Oracle Active Data Guard Oracle RAC standby

Apply Lag

Apply lag is a measure of the degree to which the data in a standby database lags behind the data in the primary database, due to delays in propagating and applying the redo.

Both Cloud Control and the DGMGRL client display the apply lag for each managed standby database. Cloud Control displays the apply lag on the Oracle Data Guard home page. The DGMGRL client displays the apply lag in the `SHOW DATABASE` output. There is no apply lag displayed for a primary database or far sync instance. For example:

```
DGMGRL> SHOW DATABASE 'South_Sales';
Database - South_Sales

Role:                PHYSICAL STANDBY
Intended State:      APPLY-ON
Transport Lag:       0 seconds (computed 0 seconds ago)
Apply Lag:          0 seconds (computed 0 seconds ago)
Apply Rate:          255.00 KByte/s
Real Time Query:    OFF
Instance(s):
  south_sales1

Database Status:
SUCCESS
```

Starting with Oracle Database Release 19c, the `SHOW CONFIGURATION LAG` command displays a summary of the broker configuration and the apply lag of all standby databases.

The apply lag can help you identify any problems that may exist with both the redo transport services and the log apply services.

You can set the `ApplyLagThreshold` database configurable property to generate a health check warning when a standby database or far sync instance lags behind the data in the primary database.

The following command sets the `ApplyLagThreshold` property to 15 seconds:

```
DGMGRL> EDIT DATABASE 'South_Sales' SET PROPERTY 'ApplyLagThreshold'=15;
Property ApplyLagThreshold updated
```

Managing Data Protection Modes

You can use the broker to set up a configuration having any of the different data protection modes.

The available modes of data protection are: maximum protection, maximum availability, and maximum performance.

This section contains the following topics to help you configure the proper protection for your configuration:

- [Setting the Protection Mode for Your Configuration](#)
- [How the Protection Modes Influence Broker Operations](#)

Setting the Protection Mode for Your Configuration

These are the steps for setting the protection mode for your configuration.

- [Setting the Protection Mode Task 1: Determine Which Data Protection Mode You Want to Use](#)
- [Setting the Protection Mode Task 2: Set up standby redo log files](#)
- [Setting the Protection Mode Task 3: Set the redo transport mode](#)
- [Setting the Protection Mode Task 4: Using DGMGRL or Cloud Control](#)

Setting the Protection Mode Task 1: Determine Which Data Protection Mode You Want to Use

Each data protection mode provides a different balance of data protection, data availability, and database performance.

To select the data protection mode that meets the needs of your business, carefully consider your data protection requirements and the performance expectations of your users.

 **Note:**

Maximum protection mode cannot be used in the following situations:

- If the only standby database in a configuration is a snapshot standby
- If a far sync instance is the only configuration member receiving redo in synchronous mode from the primary database

Maximum Availability

This protection mode provides the highest level of data protection that is possible without compromising the availability of a primary database. Transactions do not commit until all redo data needed to recover those transactions has been written to the online redo log and to the standby redo log on at least one synchronized standby database or far sync instance. If the primary database cannot write its redo stream to at least one synchronized standby database, it operates as if it were in maximum performance mode to preserve primary database availability until it is again able to write its redo stream to a synchronized standby database or far sync instance.

This mode ensures that no data loss will occur if the primary database fails, but only if a second fault does not prevent a complete set of redo data from being sent from the primary database to at least one standby database.

You can enable fast-start failover if the protection mode is maximum availability.

Maximum Performance

This protection mode provides the highest level of data protection that is possible without affecting the performance of a primary database. This is accomplished by allowing transactions to commit as soon as all redo data generated by those transactions has been written to the online log. Redo data is also written to one or more standby databases, but this is done asynchronously with respect to transaction commitment, so primary database performance is unaffected by delays in writing redo data to the standby database(s).

This protection mode offers slightly less data protection than maximum availability mode and has minimal impact on primary database performance.

This is the default protection mode.

You can enable fast-start failover if the protection mode is maximum performance.

Maximum Protection

This protection mode ensures that no data loss will occur if the primary database fails. To provide this level of protection, the redo data needed to recover a transaction must be written

to both the online redo log and to the standby redo log on at least one synchronized standby database before the transaction commits. To ensure that data loss cannot occur, the primary database will shut down, rather than continue processing transactions, if it cannot write its redo stream to at least one synchronized standby database.

Transactions on the primary are considered protected as soon as Oracle Data Guard has written the redo data to persistent storage in a standby redo log file. Once that is done, acknowledgment is quickly made back to the primary database so that it can proceed to the next transaction. This minimizes the impact of synchronous transport on primary database throughput and response time. To fully benefit from complete Oracle Data Guard validation at the standby database, be sure to operate in real-time apply mode so that redo changes are applied to the standby database as fast as they are received. Oracle Data Guard signals any corruptions that are detected so that immediate corrective action can be taken.

Because this data protection mode prioritizes data protection over primary database availability, Oracle recommends that a minimum of two standby databases be used to protect a primary database that runs in maximum protection mode to prevent a single standby database failure from causing the primary database to shut down. If only one standby database is supporting maximum protection mode, Oracle Data Guard broker will disallow the shutdown of the apply instance. This prevents the primary database from shutting down.

You can enable fast-start failover if the protection mode is maximum protection.

 **See Also:**

- [Fast-Start Failover](#) for information on fast-start failover
- *Oracle Data Guard Concepts and Administration* for complete information about data protection modes

Setting the Protection Mode Task 2: Set up standby redo log files

You must add standby redo log files on all standby databases, regardless of the protection mode you are using.

Also, Oracle requires that you add standby redo log files on the primary database in preparation for a future switchover or failover. Standby redo log files are required on the primary database if you want to enable fast-start failover.

Cloud Control automatically prompts you to select one or more standby databases in the configuration and sets up standby redo log (SRL) files on them and on the primary database in preparation for a future role change.

 **See Also:**

If you are using the DGMGRL command-line interface, follow the instructions in *Oracle Data Guard Concepts and Administration* to configure standby redo log files.

Setting the Protection Mode Task 3: Set the redo transport mode

If the data protection mode requires that you change the redo transport mode used by any of the standby databases, then either change the `LogXptMode` database property on each standby database, or set the `RedoRoutes` property on the primary database or on the far sync instance that is directly connected to the standby database.

See [Managing Redo Transport Services](#) for more information about setting the redo transport service. [Table 4-2](#) shows the protection modes and the corresponding redo transport service.

Cloud Control automatically specifies the correct redo transport service on the primary database in preparation for a future switchover.

Table 4-2 Oracle Data Guard Protection Modes and Requirements

Protection Mode	Redo Transport	Standby Redo Log Files Needed?	Usable with Fast-Start Failover?
MAXPROTECTION	SYNC	Yes	Yes
MAXAVAILABILITY	SYNC, FASTSYNC	Yes	Yes ¹
MAXPERFORMANCE	ASYNC	Yes	Yes

¹ Because FASTSYNC transport mode uses the NOAFFIRM attribute of the `LOG_ARCHIVE_DEST_n` parameter, data loss is possible. This means that a fast-start failover cannot be initiated when FASTSYNC is used and the standby is missing redo data.

Setting the Protection Mode Task 4: Using DGMGRL or Cloud Control

These steps describe how to set the protection mode using DGMGRL commands or Cloud Control.

With DGMGRL:

1. Use the `EDIT DATABASE (property)` command and specify the standby database whose redo transport service should be changed to correspond to the protection mode you plan to set. For example, if you plan to set the overall Oracle Data Guard configuration to operate in maximum availability mode, you must use the `EDIT DATABASE` command to set the `SYNC` mode for redo transport services. For example:

```
DGMGRL> EDIT DATABASE 'South_Sales' SET PROPERTY LogXptMode='SYNC';
```

Do this also for the primary database or another standby database in the configuration to ensure that it can support the chosen protection mode after a switchover.

You could also use the `RedoRoutes` property, as follows:

```
EDIT DATABASE 'North_Sales' SET PROPERTY RedoRoutes = '(LOCAL : South_Sales SYNC)';
```

2. Use the `EDIT CONFIGURATION SET PROTECTION MODE AS protection-mode` command to set the overall configuration protection mode. For example:

```
DGMGRL> EDIT CONFIGURATION SET PROTECTION MODE AS MAXAVAILABILITY;
```

See [Scenario 4: Setting the Configuration Protection Mode](#) for a DGMGRL scenario showing how to set the protection mode.

With Cloud Control:

1. On the Oracle Data Guard overview page, click the link to the right of the Protection Mode label.
2. Select Maximum Protection, Maximum Availability, or Maximum Performance and click Continue.
3. If prompted, log in to the database with `SYSDG` or `SYSDBA` privileges and click Login.
4. Select one or more standby databases to support the protection mode that you selected. If standby redo log files are needed, verify the names of the log files. Click OK.
5. On the Confirmation page, click Yes.

The broker does not allow the protection mode to be directly upgraded from maximum performance mode to maximum protection mode. You must first change from maximum performance to maximum availability, and then to maximum protection.

How the Protection Modes Influence Broker Operations

These topics describe how an Oracle Data Guard configuration's protection mode and redo transport services can affect operations such as switchovers, failovers, and disabling or enabling the configuration.

This section This section contains the following sections:

- [Upgrading or Downgrading the Current Protection Mode](#)
- [Switchover Operations](#)
- [Failover Operations](#)
- [Disable and Enable Operations](#)
- [Requirements For Removing a Database from the Configuration](#)
- [Requirements On Other Operations](#)

Upgrading or Downgrading the Current Protection Mode

No restart is necessary when you upgrade the current Oracle Data Guard protection mode to maximum availability or when you downgrade the current Oracle Data Guard protection mode.

Follow these recommendations when upgrading or downgrading the Oracle Data Guard protection mode:

- When upgrading the protection mode, upgrade the redo transport service before you upgrade the overall protection mode. At the time when you change the protection mode or reset the redo transport service of a standby database, the broker verifies that there is at least one standby database in the configuration that can support the desired grade of protection. If not, then the broker does not change the protection mode and returns an error.
- When downgrading the protection mode, downgrade the protection mode first and then change the redo transport service (if necessary). The broker will disallow a change of the redo transport service if doing so invalidates the current overall protection mode.

If you upgrade the protection mode from the maximum performance mode, the broker ensures that there is at least one standby database that receives redo via the `SYNC` transport, either directly or through a far sync instance. Additionally, for upgrades to

maximum protection mode, the broker ensures there are no gaps in the redo data on the standby database. If there are no standby databases in the configuration that meet these requirements, the request to upgrade the protection mode is rejected with an error.

Starting with Oracle Database Release 21c, you can upgrade the protection mode to maximum availability even if the primary does not have any `SYNC` standbys.

The protection mode cannot be changed if fast-start failover is enabled. An exception to this is that a downgrade to maximum availability mode is allowed when fast-start failover has been enabled in maximum protection mode.

Switchover Operations

A switchover does not change the overall Oracle Data Guard protection mode. The protection mode remains the same as it was before the switchover.

This requires that there be a standby database that is properly configured to support the current protection mode once the switchover completes. This can be either another standby database in the configuration or the current primary database that will become a standby database after the switchover completes.

Before you perform a switchover, if necessary you can add standby redo log files and set the redo transport properties on the current primary database, or on another standby database in the configuration, to the transport mode that is required to support the Oracle Data Guard protection mode. Then, when the switchover begins:

- The broker verifies the presence of standby redo log files and the redo transport service setting on each standby database and on the current primary database.
- The broker verifies there are no gaps in the redo data present on the target standby database.

If the verification is successful, the switchover continues; otherwise, the switchover fails, and the database roles and the broker configuration files remain unchanged.

WARNING:

- If the target of the switchover is a physical standby database, then the broker shuts down and restarts the primary database.

See Also:

[Switchover](#) for more information about switchovers

Failover Operations

After you perform a manual failover, the Oracle Data Guard protection mode is downgraded to maximum performance mode if the protection mode was at maximum protection. You can upgrade the protection mode later, if necessary. If the protection mode was at maximum availability or maximum performance, it remains unchanged. The redo transport services of the standby databases remain unchanged.

If fast-start failover occurs, the broker preserves the protection mode that was in effect just prior to the fast-start failover. If the protection mode was maximum protection, then the configuration protection mode is preserved, but the new primary database is set to maximum availability to allow the instance to open. When a standby becomes available that supports maximum protection mode (either because the old primary database was reinstated or due to the presence of another standby in the configuration), the database protection mode is elevated to match the configuration protection mode of maximum protection.



See Also:

[Manual Failover](#) and [Fast-Start Failover](#) for more information about manual failover and fast-start failover, respectively

Disable and Enable Operations

When you disable broker management of a standby database, the broker checks to see if the overall protection mode can still be satisfied by any of the remaining standby databases. If not, the broker rejects the disable operation. Otherwise, the broker allows the disable operation to proceed as long as fast-start failover is not enabled. If it is enabled, the broker allows the disable operation to proceed only if the standby database is not the target standby database for fast-start failovers.



WARNING:

If you disable broker management of a standby database in the broker configuration, that standby database cannot be used by the broker as a failover target in the event of loss of the primary database.

As long as fast-start failover is not enabled, you can disable the entire configuration regardless of the protection mode. You cannot disable the configuration if fast-start failover is enabled. See [Restrictions When Fast-Start Failover is Enabled](#) for more information.

When enabling broker management of the entire configuration, the broker first checks to see if the protection mode will be satisfied by the redo transport settings of the standby databases that will be enabled. If not, the enable operation fails and the configuration remains disabled. Otherwise, the enable operation successfully enables the configuration, and the broker enables the database using the settings saved in the broker configuration file.

Requirements For Removing a Database from the Configuration

When removing a standby database from the broker configuration, the broker checks to see if the protection mode will still be satisfied. The operation fails if:

- Removing the database compromises the protection mode
- Fast-start failover is enabled and you try to remove the standby database that is the target of the fast-start failover

- The configuration member to be removed has its `RedoRoutes` configurable property set to a non-null value

You can remove the configuration at any time, unless fast-start failover is enabled.

Requirements On Other Operations

Some operations that take place in a broker configuration, especially operations related to redo transport services, can affect the overall protection mode. These operations include:

- Stopping redo transport services on the primary database
- Stopping redo transport services to individual standby databases
- Downgrading the redo transport mode from `SYNC` to `ASYNC` to the only standby database that supports a configuration operating in maximum availability mode or maximum protection mode

Before any of these operations can proceed, the broker checks to see if the protection mode will be supported by the redo transport service settings on the standby databases after the operation completes. If not, the broker fails the operation and returns an error.

Managing Far Sync Instances

An Oracle Data Guard far sync instance is a redo transport destination that accepts redo from a primary database and forwards that redo to one or more redo destinations in the configuration.

It is similar to a physical standby database in that it has a control file, receives redo into Standby Redo Log files (SRLs), and archives those SRLs to local Archived Redo Logs (ARLs). But unlike a standby database, a far sync instance does not have data files, cannot be opened, and cannot apply received redo. These limitations yield the benefit of using fewer disk and processing resources. More importantly, a far sync instance provides the ability to failover to a terminal database with no data loss if it receives redo data using synchronous transport mode and the configuration protection mode is set to maximum availability.

Starting with Oracle Database Release 21c, you can use the `CREATE FAR_SYNC` command to create a new far sync instance and add it to the broker configuration. See [CREATE FAR_SYNC](#).

The following example shows how to add a far sync instance to a broker configuration.

```
DGMGRL> ADD FAR_SYNC FS1 AS CONNECT IDENTIFIER IS FS1.example.com;
Far Sync FS1 added
DGMGRL> ENABLE FAR_SYNC FS1;
Enabled.
DGMGRL> SHOW CONFIGURATION;
```

```
Configuration - DRSolution
```

```
Protection Mode: MaxPerformance
Members:
North_Sales - Primary database
FS1         - Far Sync
South_Sales - Physical standby database
```

```
Fast-Start Failover: DISABLED
Configuration Status:
SUCCESS
```

After a far sync instance has been added to the configuration, set up redo transport to support maximum availability and then upgrade the protection mode:

```
DGMGRL> EDIT DATABASE 'North_Sales' SET PROPERTY 'RedoRoutes' = '(LOCAL : FS1
SYNC)';
DGMGRL> EDIT FAR_SYNC 'FS1' SET PROPERTY 'RedoRoutes' = '(North_Sales :
South_Sales ASYNC)';
DGMGRL> EDIT CONFIGURATION SET PROTECTION MODE AS MaxAvailability;
DGMGRL> SHOW CONFIGURATION;
```

Configuration - DRSolution

```
Protection Mode: MaxAvailability
Members:
North_Sales - Primary database
  FS1       - Far Sync
  South_Sales - Physical standby database
```

```
Fast-Start Failover: DISABLED
Configuration Status:
SUCCESS
```

To ensure that maximum availability protection mode can be maintained when `South_Sales` is the primary database, after a switchover or a failover, add a second far sync instance to the configuration so that `South_Sales` can send redo in synchronous mode which in turn will send redo to the new terminal database, `North_Sales`, after the role transition.

The following example shows how to add a second far sync instance to the broker configuration:

```
DGMGRL> ADD FAR_SYNC FS2 AS CONNECT IDENTIFIER IS FS2.example.com;
Far Sync FS2 added
DGMGRL> EDIT FAR_SYNC 'FS2' SET PROPERTY 'RedoRoutes' = '(South_Sales :
North_Sales ASYNC)';
DGMGRL> ENABLE FAR_SYNC FS2;
Enabled.
DGMGRL> EDIT DATABASE 'South_Sales' SET PROPERTY 'RedoRoutes' = '(LOCAL : FS2
SYNC)';
DGMGRL> SHOW CONFIGURATION;
```

Configuration - DRSolution

```
Protection Mode: MaxAvailability
Members:
North_Sales - Primary database
  FS1       - Far Sync
  South_Sales - Physical standby database
  FS2       - Far Sync (inactive)
```

```
Fast-Start Failover: DISABLED
Configuration Status:
SUCCESS
```

If a far sync instance is monitored for availability by Oracle Clusterware (for example, in an Oracle Restart, Oracle Real Application Clusters (Oracle RAC), or Oracle RAC One Node installation), then use the `SRVCTL` utility to specify a default open mode of mount. You can use a command such as the following:

```
srvctl modify database -d <db_unique_name> -startoption MOUNT
```

 **See Also:**

- *Oracle Data Guard Concepts and Administration* for more information about creating a far sync instance and redo transport

Managing Fast-Start Failover

You can enable **fast-start failover** to allow the broker to determine if a failover is necessary and to initiate a failover to a standby database from a list of one or more pre-specified target standby databases.

The failover can be set up for either no data loss or a configurable amount of data loss. In addition, you can specify under which conditions or errors you want a failover to be initiated. Oracle also provides the `DBMS_DG` PL/SQL package to allow an application to request a fast-start failover.

You use broker configuration properties to control the behavior of fast-start failover. You can also use Cloud Control or the `DGMGRL ENABLE FAST_START FAILOVER CONDITION` and `DISABLE FAST_START FAILOVER CONDITION` commands to specify conditions for which a fast-start failover should occur.

Configure Properties to Tune Fast-Start Failover

You can set various properties to tune how fast-start failover behaves.

The configurable properties for fast-start failover include:

- `FastStartFailoverThreshold`

Set the `FastStartFailoverThreshold` configuration property to specify the number of seconds you want the observer and target standby database to wait (after detecting the primary database is unavailable) before initiating a failover. See [Enabling Fast-Start Failover](#) for more information and an example.

- `FastStartFailoverPmyShutdown`

The `FastStartFailoverPmyShutdown` configuration property controls whether the primary database will shut down if redo generation has been stalled (`FS_FAILOVER_STATUS` column of `V$DATABASE` contains a value of `STALLED`) and the primary database has lost connectivity with the observer and target standby database for longer than the number of seconds specified by the `FastStartFailoverThreshold` configuration property. The default value for `FastStartFailoverPmyShutdown` is `TRUE`.

 **Note:**

The primary database is always shut down if a user configurable fast-start failover condition is detected or if an application initiated a fast-start failover by calling the `DBMS_DG.INITIATE_FS_FAILOVER` function.

- `FastStartFailoverLagLimit`

The fast-start failover feature can be configured on databases operating in maximum performance mode. Destinations that receive redo in `ASync` mode will be acceptable fast-start failover target standby databases, and these destinations can lag the primary in terms of redo received and applied. A configurable time-based limit can be specified through the `FastStartFailoverLagLimit` configuration property. If the standby database's applied redo point is within this many seconds of the primary's redo generation point, a fast-start failover will be allowed. If its applied point lags beyond this limit, a fast-start failover is not allowed.

The `FastStartFailoverLagLimit` configuration property can also be used if fast-start failover is enabled when the configuration is operating in maximum availability mode. It cannot be used when the configuration is operating in maximum protection mode. When `FastStartFailoverLagLimit` is set to a non-zero value and the configuration is operating in maximum availability mode, a zero data loss failover or a data loss failover is possible. If a data loss failover is performed, the amount of data loss will not exceed the number of seconds specified by the `FastStartFailoverLagLimit` configuration property. Note that the redo transport mode of the target standby must be set to the value of `Sync` or `FASTSync` or `ASync` when a non-zero value is specified for the `FastStartFailoverLagLimit` property and the protection mode is maximum availability mode or maximum performance mode. If you want to change protection mode or redo transport mode to `Sync` or `FASTSync`, you must first disable fast-start failover. Likewise, changing the protection mode from maximum availability mode to maximum performance mode will require first disabling fast-start failover. Reinstatement of an old primary will be possible after a fast-start failover to a target standby. If the observer rediscovers the old primary, it will automatically reinstate the old primary and any redo generated within the specified lag will be lost.



See Also:

[Oracle Data Guard Broker Properties](#) for more information

- `FastStartFailoverAutoReinstate`

The `FastStartFailoverAutoReinstate` configuration property controls whether the former primary database is automatically reinstated if a fast-start failover occurred because the primary database crashed or was stalled for longer than `FastStartFailoverThreshold` seconds. The default value for `FastStartFailoverAutoReinstate` is `TRUE`.

If you want to perform diagnostic or repair work after failover has completed, you can avoid an automatic reinstatement by setting the `FastStartFailoverAutoReinstate` configuration property to `FALSE`.



Note:

The former primary database is never automatically reinstated if a fast-start failover occurred because a user configurable fast-start failover condition was detected or because an application initiated a fast-start failover by calling the `DBMS_DG.INITIATE_FS_FAILOVER` function.

- `FastStartFailoverTarget`

The `FastStartFailoverTarget` configuration property specifies the `DB_UNIQUE_NAME` values of the databases that are eligible to be targets of a fast-start failover when this database is the primary database.
- `ObserverPingInterval`

The `ObserverPingInterval` configuration property specifies how frequently the observer must ping the primary database. This property is measured in milliseconds. The minimum value is 100 milliseconds. To achieve lower detection times for primary database failures, you must set the `ObserverPingInterval` and `ObserverPingRetry` properties before enabling fast-start failover.
- `ObserverPingRetry`

The `ObserverPingRetry` configuration property specifies the number of times that the observer retries a failed ping before it initiates a failover to the target standby database. A failed ping is a ping to the primary database that failed or took longer than the time specified by the `ObserverPingInterval` property. You must set both the `ObserverPingRetry` and `ObserverPingInterval` properties to achieve lower detection times for primary database failures. The minimum value is 10.
- `ObserverReconnect`

The `ObserverReconnect` configuration property specifies how often the observer establishes a new connection to the primary database. When this property is set to the default value of 0, it prevents the observer from periodically establishing a new connection with the primary database. While this eliminates the processing overhead associated with periodically establishing a new observer connection to the primary database, it also prevents the observer from detecting that it is not possible to create new connections to the primary database. Oracle recommends that this property be set to a value that is small enough to allow timely detection of faults at the primary database, but large enough to limit the overhead associated with periodic observer connections to an acceptable level.
- `ObserverOverride`

The `ObserverOverride` configuration property, when set to `TRUE`, allows an automatic failover to occur when the observer has lost connectivity to the primary, even if the standby has a healthy connection to the primary.

Configure Conditions for Fast-start Failover

By default, a fast-start failover is done when neither the observer nor the standby can reach the primary after the configured time threshold (`FastStartFailoverThreshold`) has passed.

There are also other conditions under which you might want a fast-start failover to occur.

The configurable conditions fall into two classes: those detected through the database health-check mechanism and those detected through errors raised by the Oracle server (such as ORA errors). When a specified condition occurs, the observer will initiate a fast-start failover without waiting for `FastStartFailoverThreshold` to expire, assuming the standby is in a valid state to accept a failover.

Each condition may be enabled or disabled individually. The Oracle Data Guard configuration persists all user specified configurable fast-start failover conditions in the broker configuration file.

The observer will detect when the primary database has signaled any of the enabled health-check conditions and will immediately initiate a fast-start failover, assuming the standby is in a valid fast-start failover state (observed and either synchronized or within lag limits) to accept a failover.

For specified Oracle ORA-Error conditions, the primary database will notify the observer if the error is signaled and the observer will immediately initiate a fast-start failover, assuming the standby is in a valid fast-start failover state (observed and either synchronized or within lag limits) to accept a failover. Please note that the only Oracle ORA-Error for which fast-start failover can be triggered is ORA-240.

 **Note:**

The primary database will shut down and the observer will not attempt to automatically reinstate the former primary database.

 **See Also:**

- Cloud Control online help
- [ENABLE FAST_START FAILOVER CONDITION](#)
- [DISABLE FAST_START FAILOVER CONDITION](#)

Application Initiated Fast-Start Failover

You can use the `DBMS_DG` PL/SQL package to allow an application to direct a fast-start failover when it encounters specific conditions.

See "[Directing a Fast-Start Failover From an Application](#)".

Managing Database Conversions

You can use the `DGMGRL CONVERT DATABASE` command to convert a physical standby database to a snapshot standby database.

A snapshot standby database is a fully updatable standby database.

Like a physical or logical standby database, a snapshot standby database receives and archives redo data from a primary database. However, unlike a physical or logical standby database, a snapshot standby database does not apply the redo data that it receives. The redo data received by a snapshot standby database is not applied until the snapshot standby is converted back into a physical standby database, after first discarding any local updates made to the snapshot standby database.

To convert a physical standby database to a snapshot standby database you must have Flashback Database enabled. The following example shows how to convert a physical standby database to a snapshot standby database:

```
DGMGRL> CONVERT DATABASE 'South_Sales' TO SNAPSHOT STANDBY;
```

When you are ready to convert the snapshot back into a physical standby, use the DGMGRL `CONVERT DATABASE` command as follows:

```
DGMGRL> CONVERT DATABASE 'South_Sales' TO PHYSICAL STANDBY;
```

Database Status

In general, the broker checks the health of a database by verifying that the actual database state and settings match those described in the broker configuration file.

This is done by checking if any component of the Oracle Data Guard configuration is functioning incorrectly (for example, if redo transport services have an error), and by checking if other required database settings are correctly set (for example, if the server parameter files are available and if the `ARCHIVELOG` mode is turned on). The following is a detailed list of what is being checked by the broker on a primary database and a standby database.

On a primary database, the health check determines whether the following conditions are met:

- Database is in the state specified by the user, as recorded in the broker configuration file
- Database is in the correct data protection mode
- Database is using a server parameter file
- Database is in the `ARCHIVELOG` mode
- Database guard is turned off
- Supplemental logging is turned on when there is a logical standby database in the configuration
- Redo transport services do not have any errors
- Database settings match those specified by the broker configurable properties
- Redo transport settings match those specified by the redo transport-related properties of the standby databases
- Current data protection level is consistent with configured data protection mode
- Primary database is able to resolve all gaps for all standby databases

On a standby database, the health check determines whether the following conditions are met:

- Database is in the state specified by the user, as recorded in the broker configuration file
- Database is using a server parameter file
- Database settings match those specified by the broker configurable properties
- Database guard is turned on when the database is a logical standby database
- Primary and target standby databases are synchronized or within lag limits if fast-start failover is enabled

Querying Database Status

Certain monitorable properties can be used to query the database status.

The following properties are directly accessed through the DGMGRL command-line interface:

- LogXptStatus
- InconsistentLogXptProps

 **Note:**

Cloud Control rearranges the values of these properties for presentation in the GUI.

You can use the `SHOW DATABASE <db_unique_name>` command to get a brief description of the database (name, role, and so on), database status, and information about any health check problems. For example, the output of the following `SHOW DATABASE` command shows two problems: some redo transport services errors and an inconsistent redo transport-related property

```
DGMGRL> SHOW DATABASE 'North_Sales';

Database - North_Sales
  Role: PRIMARY
  Intended State: TRANSPORT-OFF
  Instance(s):
    north_sales1
      Error: ORA-16737: the redo transport service for standby
        database "South_Sales" has an error

    north_sales2
      Error: ORA-16737: the redo transport service for standby
        database "South_Sales" has an error
      Warning: ORA-16715: redo transport-related property
        ReopenSecs of standby "South_Sales" is inconsistent
  Database Status:
  ERROR
```

To further check the details about the database status, you can use the following monitorable properties:

- `LogXptStatus` — lists all log transport errors detected on all instances of the primary database.
- `InconsistentLogXptProps` — lists all redo transport-related properties of standby databases that have inconsistent values between the broker configuration file and the redo transport settings.

Issue the following `SHOW DATABASE` commands to obtain further details about the problems.

```
DGMGRL> SHOW DATABASE 'North_Sales' 'LogXptStatus';
LOG TRANSPORT STATUS
PRIMARY_INSTANCE_NAME  STANDBY_DATABASE_NAME  STATUS
      north_sales1      South_Sales  ORA-12541: TNS:no listener
      north_sales2      South_Sales  ORA-12541: TNS:no listener

DGMGRL> SHOW DATABASE 'North_Sales' 'InconsistentLogXptProps';
INCONSISTENT LOG TRANSPORT PROPERTIES
INSTANCE_NAME  STANDBY_NAME  PROPERTY_NAME  MEMORY_VALUE  BROKER_VALUE
      north_sales2  South_Sales  ReopenSecs      600           300
```

**See Also:**

[Oracle Data Guard Broker Properties](#) for detailed information about database properties

Validating a Database Before a Role Change

You can use the `VALIDATE DATABASE` command to perform a comprehensive set of database checks prior to performing a role change.

The command checks the following items:

- Whether there is missing redo data on a standby database
- Whether flashback is enabled
- The number of temporary tablespace files configured
- Whether an online data file move is in progress
- Whether online redo logs are cleared for a physical standby database
- Whether standby redo logs are cleared for a primary database
- The online log file configuration
- The standby log file configuration
- Apply-related property settings
- Transport-related property settings
- Whether there are any errors in the Automatic Diagnostic Repository (for example, control file corruptions, system data file problems, user data file problems)

**See Also:**

- ["VALIDATE DATABASE"](#) for a description of the command and for examples that show command output for various scenarios

Validating the Server Parameter Files Before a Role Change

Use the `VALIDATE DATABASE SPFILE` command to compare the contents of the server parameter file (SPFILE) between the primary and standby database.

Comparing the primary to the standby lets you determine whether there are any missing parameters in either database's SPFILE, or whether the entries contain different values.

**See Also:**

- [VALIDATE DATABASE SPFILE](#) for a description of the command and for examples that show command output

Validating the Network Configuration Before a Role Change

Use the `VALIDATE NETWORK CONFIGURATION` command to perform network connectivity checks between members of a configuration.

Performing a network connectivity check identifies potential network configuration problems before a role change is attempted.

See Also:

- [Validating the Network Configuration Before a Role Change](#)

Validating the Static Connect Identifier Before a Role Change

A single-instance database on which Oracle Restart is not configured must have a static service registered with the listener so that the DGMGRL CLI can automatically start the instance when necessary (for example, for the new standby after a switchover).

The broker sets up a default value for the `StaticConnectIdentifier` property that uses that static service (assuming the default value of `db_unique_name_DGMGRL` is used for the static service name). This connect identifier is used for instance restart.

Use the `VALIDATE STATIC CONNECT IDENTIFIER` command to confirm that the connect identifier specified by the `StaticConnectIdentifier` property can be used to restart an instance. The command does not restart the database, but rather checks that the service specified in the connect identifier is registered with the listener.

See Also:

- [VALIDATE STATIC CONNECT IDENTIFIER](#) for a description of the command and for examples that show command output

Validating the DGConnectIdentifier Property

The DGMGRL command `VALIDATE DGConnectIdentifier` enables users to check to see whether a connection string is valid for the `DGConnectIdentifier` property or not..

The `VALIDATE DGConnectIdentifier` command checks if it is able to be translated to something useful and makes a connection using it on all configuration members. This can also be used prior to adding a member to the configuration. If no configuration exists, the command checks a connection string at the database DGMGRL is connected to. If a configuration exists, all members use the specified connection string to see that the connection works.

This command performs the following for each instance of all members.

- Prints a network translation of the connection string at the instance.

- Prints environment variables related to network configuration at the instance.
- Makes a new connection using the translated network address at the instance.
- If a connection test succeeds, the instance name and `db_unique_name` of the connected database will be printed.



See Also:

- [VALIDATE DGConnectIdentifier](#) for a description of the command and for examples that show command output

5

Managing Members of a DG PDB Broker Configuration

Learn how to manage DG PDB broker configurations and monitor the state of configuration members.



Note:

The only supported protection mode for DB PDB configurations is maximum performance mode.

This chapter contains the following topics:

- [Managing Broker Configuration Members](#)
- [Managing States of Broker Configuration Members](#)
- [State Transitions for DG PDBs](#)
- [Managing Database Properties in DG PDB Configurations](#)
- [Managing Redo Transport Services for DG PDBs](#)
- [Viewing Transport Lag and Apply Lag](#)
- [Monitoring a DG PDB Configuration](#)
- [Querying Database Status in DG PDB Configurations](#)

Managing Broker Configuration Members

The broker uses information in its configuration files to manage and monitor the state of members in the DG PDB configuration.

The `COMPATIBLE` initialization parameter in the source database and the target database must be set to 21.1 or higher.

Tasks performed for CDB configurations such as setting configuration properties, creating configuration files, starting Data Guard broker, and enabling or disabling broker configurations are also applicable to DG PDB configurations.

 **Note:**

Although you can configure a DG PDB for a PDB that is part of a CDB that already has a standby database, a role change between such a PDB and its DG PDB results in an inconsistent PDB configuration between the primary and standby databases. Similarly, a role change between the primary and standby database will result in the DG PDB being orphaned because it no longer has a source PDB. When you use Data Guard broker to manage the Data Guard primary and standby databases, creating a DG PDB in such a scenario not allowed.

When broker is not used to manage the Data Guard environment, it is recommended to avoid setting up a PDB for Data Guard protection when the container to which the PDB belongs already has a Data Guard standby database.

Managing States of Broker Configuration Members

The standby PDBs in a DG PDB configuration can be in the `APPLY-ON` or `APPLY-OFF` state.

The following table describes the various states for standby PDBs in a DG PDB configuration. No standby databases are listed because they cannot be used in a DG PDB configuration.

Table 5-1 Database States and Descriptions in DG PDB Configurations

Database role	State Name	Description
Target PDB	<code>APPLY-ON</code>	Redo Apply is started on the target PDB within the target configuration. If this is an Oracle Real Application Clusters (Oracle RAC) instance, redo apply is started only on the instance where the command is run. The PDB cannot be opened if recovery is running. If the PDB is open when the command to start recovery is run, the PDB is closed.
Target PDB	<code>APPLY-OFF</code>	Redo Apply is stopped on the target PDB.

State Transitions for DG PDBs

Use the `DGMGRL EDIT DATABASE` command to explicitly change the state of a standby PDB. You can only change the state of the entire database, including all its PDBs.

For example, the following command changes the state of the target database `target-prim` to `APPLY-OFF`:

```
DGMGRL > EDIT PLUGGABLE DATABASE 'target-prim' SET STATE = 'APPLY-OFF';  
Succeeded.
```

Managing Database Properties in DG PDB Configurations

Database properties of members in a DG PDB configuration are of two types: monitorable properties or configurable properties.

Use monitorable properties to view run-time information related to configuration members. Monitorable properties can be viewed only when the member is enabled. Use the `SHOW DATABASE VERBOSE` command to list all monitorable properties. Use the `SHOW DATABASE` command to view details about a particular property. Use the `SHOW PLUGGABLE DATABASE` command to view information about the source PDB or target PDB.

Configurable properties impact the operation or configuration of a database. Use configurable properties to view and dynamically update information related to configuration members. You can edit properties regardless of the member's state. If the member is disabled when a property is edited, the property value takes effect after the member or configuration is enabled, as appropriate.

Use the `EDIT DATABASE` command to update the properties of the source database or target database. Use the `EDIT PLUGGABLE DATABASE` command to update the state of a source PDB or target PDB.

The `SHOW DATABASE` command has been enhanced to display information about Data Guard at the PDB level. In a DG PDB configuration, the `SHOW DATABASE` output shows whether the database has any source or target PDBs, and the number of PDBs in each role.

Example 5-1 Displaying the Properties of a Source Database

```
DGMGRL> SHOW DATABASE boston;
```

```
Database - boston  
  Role: PRIMARY  
  Intended State: TRANSPORT-ON  
  PDB Data Guard Role: SOURCE and TARGET  
  Instance(s):  
    bos1  
  Data Guard Source PDB(s): 1  
  Data Guard Target PDB(s): 2
```

```
Database Status:  
SUCCESS
```

Example 5-2 Displaying the Properties of Individual PDBs

This example uses the `SHOW PLUGGABLE DATABASE` command to display information about the target PDB `dgpdb_sales`.

```
DGMGRL> SHOW PLUGGABLE DATABASE 'dgpdb_sales' AT 'newyork';
```

```
Pluggable database 'dgpdb_sales' at database 'newyork'
```

```
Data Guard Role: Physical Standby  
Con_ID: 7  
Source: con_id 6 at boston  
Transport Lag: +00 00:00:00 seconds  
Intended State: APPLY-ON  
Apply State: Running
```

Managing Redo Transport Services for DG PDBs

The broker automatically starts redo transport services between a source database and target database once you set up at least one PDB for replication.

To turn on redo transport services for the source database:

1. Connect to the source database.
2. Use the `EDIT CONFIGURATION SET AS SOURCE` command to specify the source database that must be set up for redo transport.

Viewing Transport Lag and Apply Lag

Transport lag is a measure of the degree to which redo transport or redo apply lags behind the source database.

The redo transport service transmits redo data from the source database to the target database. The log apply services at the individual target PDB updates the target PDB with this redo data, thereby keeping the target PDBs consistent with the source PDBs. Each target PDB has its own log apply service. The archived redo log files and standby redo log files contain all of the database changes except for unrecoverable or unlogged changes.

Transport Lag

The transport lag is displayed in the output of the `SHOW PLUGGABLE DATABASE` command. This lag represents, in seconds, the amount of redo data that has not been received by the target CDB for the specified DG PDB.

Average Apply Rate

If the Apply State is Running, the average apply rate is displayed in the output of the `SHOW PLUGGABLE DATABASE` command. The average apply rate pertains to a particular PDB.

Monitoring a DG PDB Configuration

Monitor the status of the configuration and its members to view statistics, diagnostic information, detect problems, and quickly take necessary actions.

After a DG PDB configuration is enabled, the broker schedules an automatic health check on the individual members. It monitors and reports problems with redo transport and recovery progress. Diagnostic information is written to the broker log files (drc*.log) and can be viewed using DGMGRL commands.

Querying Database Status in DG PDB Configurations

Use DGMGRL to query information about the DG PDB configuration and its members.

You can view the status of the DG PDB configuration and detailed information about the source database and target database.

Example 5-3 Displaying Information for a DG PDB Configuration

This example displays information about a DG PDB configuration.

```
DGMGRL> SHOW CONFIGURATION;
```

```
Configuration - MyConfig1
  Protection Mode: MaxPerformance
  Members:
    boston - Primary database
    newyork - Primary database in MyConfig2 configuration

Data Guard for PDB: Enabled in SOURCE role

Configuration Status:
SUCCESS
```

6

Switchover and Failover Operations

Learn how to use Oracle Data Guard broker to manage databases during switchover and failover.

- [Overview of Switchover and Failover in a Broker Environment](#)
- [Choosing a Target Standby Database](#)
- [Switchover](#)
- [Manual Failover](#)
- [Fast-Start Failover](#)
- [Database Client Considerations](#)

Overview of Switchover and Failover in a Broker Environment

Oracle Data Guard helps you change the role of databases between primary and standby using either a switchover or failover operation.

- A **switchover** is a role reversal between the primary database and one of its standby databases. A switchover guarantees no data loss and is typically done for planned maintenance of the primary system. During a switchover, the primary database transitions to a standby role, and the standby database transitions to the primary role.
- A failover is a role transition in which one of the standby databases is transitioned to the primary role after the primary database (all instances in the case of an Oracle RAC database) fails or has become unreachable. A failover may or may not result in data loss depending on the protection mode in effect at the time of the failover.

The broker simplifies switchovers and failovers by allowing you to invoke them using a single key click in Oracle Enterprise Manager Cloud Control (Cloud Control) or a single command in the DGMGRL command-line interface (referred to in this documentation as *manual* failover). As part of managing the switchover, the broker automatically does the following:

- Sets up redo transport from the new primary to the other members of the configuration
- Starts Redo Apply services on the new standby
- Ensures the other standbys in the broker configuration are viable to the new primary
- Integrates with Oracle Clusterware and Oracle Global Data Services (GDS) to ensure that the proper services are started after a role change

When fast-start failover is enabled, the broker determines if a failover is necessary and initiates the failover to the current target standby database automatically, with no need for manual intervention. The reduced need for manual intervention can increase availability without increasing management costs. Manual failover gives you control over exactly when a failover occurs and to which target standby database. Regardless of the method you choose, the broker coordinates the role transition on all databases in the configuration.

Choosing a Target Standby Database

When you select a standby database to be the next primary database after a switchover or a failover, there are several factors to consider.

You need to consider all of the options at the time you are building your Oracle Data Guard configuration, including factors such as the characteristics of physical standbys versus logical standbys versus snapshot standbys, the network latency to your standby database sites, the computing capabilities at a future primary database site, and so on.

Note:

A snapshot standby cannot be the target of a switchover or fast-start failover operation. You can, however, perform a manual failover to a snapshot standby.

A far sync instance or Zero Data Loss Recovery Appliance is not a database and therefore cannot be the target of a role transition.

For switchovers, understanding all of the factors can simplify the choice of which standby database to consider as your new primary database. In disaster situations where a failover is necessary, you may be more limited as to which standby database is the best one to pick up the failed primary database's activities. [Choosing a Target Standby Database for Switchover](#) and [Choosing a Target Standby Database for Failover](#) provide guidelines to help you choose a target standby database.

Note:

For fast-start failover, you must pre-select at least one target standby database. You can also pre-select multiple targets if you want to; the selection does not have to be reciprocal.

Determining a Database's Readiness to Change Roles

To help you select an appropriate switchover or failover target, use the following DGMGRL commands which perform checks on the database to determine its readiness to complete a role change.

- [VALIDATE DATABASE](#)
- [VALIDATE NETWORK CONFIGURATION](#)
- [VALIDATE DATABASE SPFILE](#)
- [VALIDATE STATIC CONNECT IDENTIFIER](#)
- [VALIDATE DGConnectIdentifier](#)

 **See Also:**

- "[Scenario 9: Performing a Switchover Operation](#)" for an example of using the `VALIDATE DATABASE` command to show a database's readiness to complete a role switchover
- "[Scenario 10: Performing a Manual Failover Operation](#)" for an example of using the `VALIDATE DATABASE` command to show a database's readiness to complete a role failover

Choosing a Target Standby Database for Switchover

When performing a switchover in a configuration whose standby databases are all of the same type (all physical or all logical standby databases), choose the standby database that has the least amount of unapplied redo (smallest apply lag).

By choosing the standby database with the least amount of unapplied redo, you can minimize the overall time it takes to complete the switchover operation. For example:

- Using DGMGRL, you can do this by examining the output of the `SHOW CONFIGURATION LAG`.
- Using Cloud Control, you can view the value of the `ApplyLag` column for each standby database in the `Standby Databases` section of the Oracle Data Guard Overview page.

If the configuration contains both physical and logical standby databases, consider choosing a physical standby database (that has the least amount of unapplied redo) to be the target standby database. A switchover to a physical standby database is preferable because all databases in the configuration will be available as standby databases to the new primary database after the switchover operation completes. Whereas a switchover to a logical standby database will invalidate and disable all of the physical and snapshot standby databases in the configuration. You will then need to re-create the physical standby databases from a copy of the new primary database before you can reenable them. Alternatively, if you intend to switch back to the original primary relatively soon, then you may re-enable the disabled standby databases after the switch back.

You cannot perform a switchover to a snapshot standby database unless you first convert it back to a physical standby database.

 **Note:**

If the Oracle Data Guard configuration is operating in maximum protection mode, the broker does not allow a switchover to occur to a logical standby database. The configuration must be operating in either maximum availability mode or maximum performance mode in order to be able to switch over to a logical standby database.

Choosing a Target Standby Database for Failover

When performing a failover in a configuration whose standbys are all of the same type, choose the standby database that has the smallest transport lag.

Choosing the standby database with the smallest transport lag can minimize the amount of data loss and in some cases, incur no data loss at all.

If the configuration contains physical, snapshot, and logical standby databases, consider choosing a physical standby database as the target standby database. A failover to a physical standby database is preferable because it is likely that all standby databases in the configuration will still be available as standby databases to the new primary database after the failover operation completes.

You may failover to a snapshot standby database. However failing over to a snapshot standby database will require more time because the broker must first convert it back to a physical standby database. After the conversion, the broker will start Redo Apply to apply accumulated redo data, before failing the database over to the primary role. Because the broker performs the failover after converting the snapshot standby database to a physical standby database, it is likely that all standby databases in the configuration will still be available as standby databases to the new primary database after the failover operation completes.

A failover to a logical standby database requires that all physical and snapshot standby databases be re-created from a copy of the new primary database after the failover completes. In addition, a logical standby database may contain only a subset of the data present in the primary database. (For example, if the `DBMS_LOGSTDBY.SKIP` procedure was used to specify which database operations done on the primary database will not be applied to the logical standby database.)

However, there may be exceptions to the recommendation to choose a physical standby database as the target standby database. For example, if all your physical standbys are also unavailable, then failing over to a logical standby is your only choice.

Switchover

A switchover is a role reversal between the primary database and one of its standby databases.

A switchover guarantees no data loss and is typically done for planned maintenance of the primary system. During a switchover, the primary database transitions to a standby role, and the standby database transitions to the primary role.

Whenever possible, you should switch over to a physical standby database:

- If the switchover transitions a physical standby database to the primary role, then:
 - The original primary database will be switched to a physical standby role.
 - Other members of the configuration will receive redo from the designated redo source based on the new primary.
 - The original primary database will be restarted as a part of the switchover operation. Note that the new primary database does not need to be restarted.

Standby databases not involved in the switchover (known as bystander standby databases) continue operating in the state they were in before the switchover occurred and will automatically begin applying redo data received from the new primary database.

- If the switchover transitions a logical standby database to the primary role, then:
 - The original primary database will be switched to a logical standby role.

- Neither the primary database nor the logical standby database needs to be restarted after the switchover completes.

Other logical standby bystander databases in the broker configuration will remain viable after the switchover. All physical and snapshot standby databases will be disabled and must be re-created from a copy of the new primary database after a switchover to a logical standby database.

Switchover to a logical standby database is disallowed when the configuration is operating in maximum protection mode.

 **WARNING:**

Switching over to a logical standby database results in the snapshot and physical standby databases in the broker configuration being disabled by the broker, making these databases no longer viable as standby databases. [Reenabling Disabled Databases After a Role Change](#) describes how to restore their viability as standby databases.

If you intend to switch back to the original primary database relatively soon, you may allow the physical and snapshot standbys to remain disabled. Once you have completed the switchover back to the original primary, you may then reenabling the physical and snapshot standby databases since they are still viable standbys for the original primary database.

Before You Perform a Switchover Operation

These are some points to consider before you begin a switchover.

- When you start a switchover, the broker verifies that at least one standby database, including the primary database that is about to be transitioned to the standby role, is configured to support the overall protection mode (maximum protection, maximum availability, or maximum performance) after the switchover is completed.
- Prepare the primary database in advance for its possible future role as a standby database in the context of the overall protection mode (see [Managing Data Protection Modes](#)). Such preparation includes:
 - Ensuring that standby redo log files are configured on the primary database.
 - Presetting database properties related to redo transport services, such as `LogXptMode`, `NetTimeout`, `StandbyArchiveLocation`, `StandbyAlternateLocation`, and `RedoRoutes`. For more details about managing redo transport services using database properties, see [Managing Redo Transport Services](#).
 - Reset database properties related to Redo Apply services, such as `DelayMins`. Any apply delay must be removed before beginning a switchover. The broker will not allow a switchover to a standby that has an apply delay configured (`DelayMins` property is set to a non-zero value). For more details about managing Redo Apply services using properties, see [Managing Log Apply Services](#).
 - For each temporary table, verifying that temporary files associated with that table on the primary database also exist on the standby database.

Note that the broker does not use the properties to set up redo transport services and Redo Apply services until you actually switch over the primary database to the standby role. Thus, the validity of the values of these properties is not verified until after the

switchover. Once you set these properties, their values persist through role changes during switchover and failover.

- If fast-start failover is enabled, then a switchover can be performed only to the pre-specified target standby database and only if the standby database is synchronized with the primary database or is within the configured lag limit, for the max availability and max performance modes respectively. For information about enabling fast-start failover, see [Enabling Fast-Start Failover](#).
- You can use the `SHOW CONFIGURATION WHEN PRIMARY IS` command to show the redo transport configuration (based on each member's setting of the `RedoRoutes` property) that would be in effect if the specified database were the primary database. You can use this information to identify ahead of time any redo transport configurations that would be incorrect after a role change, including any standbys that will not receive redo because the `RedoRoutes` property was not configured correctly.

After a switchover completes, the broker preserves the overall Oracle Data Guard protection mode as part of the switchover process by keeping the protection mode at the same protection level (maximum protection, maximum availability, or maximum performance) it was at before the switchover. Apply services on all other bystander standby databases automatically begin applying redo data received from the new primary database.

If there are physical or snapshot standby databases in the configuration and the switchover occurs to a logical standby database, you need to re-create those databases from a copy of the new primary database and then reenables those databases, as described in [Reenabling Disabled Databases After a Role Change](#).

 **Note:**

In an Oracle Data Guard configuration, the `SRVCTL -startoption` and `-role` are updated after switchover to reflect the current open mode and database role on the new primary and standby databases. See "[Database Service Configuration Requirements](#)" for additional information about how the broker interacts with Oracle Restart.

Starting a Switchover

When a switchover is started, the primary and standby databases that are involved should have as small a redo lag as possible.

The act of switching roles should be a well-planned activity. *Oracle Data Guard Concepts and Administration* provides information about setting up the databases in preparation of a switchover.

To start a switchover using Cloud Control, select the standby database that you want to change to the primary role and click **Switchover**. When using DGMGRL, you need to issue the `SWITCHOVER` command, specifying the name of the standby database that you want to change into the primary role.

The broker controls the rest of the switchover.

How the Broker Performs a Switchover

These are the actions the broker performs after you start a switchover.

1. Verifies that the primary and the target standby databases are in the following states:
 - a. The primary database is enabled and is in the `TRANSPORT-ON` state.
 - b. The target standby database is enabled and is in the `APPLY-ON` state.

The broker allows the switchover to proceed as long as there are no errors for the primary database and the standby database that you selected to participate in the switchover operation. Errors occurring for any other configuration members will not impede the switchover.

2. If block change tracking is enabled on the primary, and the target standby database is mounted, broker remembers this setting.
3. If the `WAIT` option is included in the `SWITCHOVER` command, and the databases are managed by Oracle Clusterware:
 - The broker notifies Oracle Clusterware to stop active services.
 - The broker continuously monitors for all sessions that are connected through these services to exit or for the specified wait time expires.
 - If a non-zero value is specified for the `WAIT` option, broker waits for the amount of time specified in the `WAIT` option.
 - If no value is specified for the `WAIT` option, broker waits for the amount of time specified by maximum configured `drain_timeout` amongst the active services. The `drain_timeout` is specified in the `SRVCTL add service` command.

Note that a switchover operation may be started before the specified wait time, if all the sessions that are connected through the active services exit.

4. Switches roles between the primary and standby databases.

The broker first converts the original primary database to run in the standby role. If any errors occur during either conversion, the broker stops the switchover. See [Troubleshooting Problems During a Switchover Operation](#) for more information.

5. Updates the broker configuration file to record the change in roles.

This allows the appropriate Data Guard services, such as redo transport or redo apply, to be started when the database is restarted later for any reason.

6. If the new primary database was a primary database in the past, and had block change tracking enabled, broker enables block change tracking on the new primary.
7. Restarts the new standby (former primary) database if the switchover occurs to a physical standby database, and Redo Apply begins applying redo data from the new primary database. If this is an Oracle RAC physical standby database managed by Oracle Clusterware, then the broker directs Oracle Clusterware to restart the new standby database.

If the switchover occurs to a physical standby database, and the former primary database is managed by Oracle Clusterware, broker directs Oracle Clusterware to restart the new physical standby database. Else, broker restarts the new physical standby database.

After the restart, Redo Apply begins applying redo data from the new primary database.

8. The new primary database is opened in read/write mode and redo transport services are started.

If the former physical standby database was running with real-time query enabled, the new physical standby database will run with real-time query enabled.

If the database is managed by Oracle Clusterware, broker does not open any of the PDBs. Instead, Oracle Clusterware opens PDBs on particular instances based on the service configuration. If the database is not managed by Oracle Clusterware, broker opens all the PDBs on the new primary database and on the target standby database (if real-time query is enabled).

The broker verifies the state and status of the databases to ensure that the switchover transitioned the databases to their new role correctly. Bystander standby databases that are not disabled by the broker after the switchover will continue operating in the state they were in before the switchover. SQL Apply on all other bystander standby databases automatically begin applying redo data received from the new primary database.

In the rare event that a switchover operation fails and you are left with no primary database, retry the switchover command. You can switch back to the original primary and then either retry the switchover to the original target standby, or choose another standby in the configuration to switch over to.

Manual Failover

In a manual failover, you convert a standby database to a primary database because the original primary database failed and there is no possibility of recovering the primary database in a timely manner.

There may or may not be data loss depending upon whether your primary and target standby databases were synchronized at the time of the primary database failure. The word *manual* is used to contrast this type of failover with a fast-start failover (described in [Fast-Start Failover](#)).



Note:

You can perform a manual failover even if fast-start failover is enabled. See [Performing Manual Role Changes When Fast-Start Failover Is Enabled](#) for more information.

The following sections describe how to perform manual failovers:

- [Complete and Immediate Manual Failovers](#)
- [Performing a Manual Failover Operation](#)
- [Reenabling Disabled Databases After a Role Change](#)

Complete and Immediate Manual Failovers

You can use Cloud Control or DGMGRL, to perform either a *complete* (recommended) or an *immediate* failover.

- A *complete* failover is the recommended and default failover option. It automatically recovers the maximum amount of redo data for the protection mode the configuration is operating in. A complete failover also attempts to avoid disabling any standby databases that were not the target of the failover, so that they may continue serving as standby databases to the new primary database.

Whether or not standby databases that were not the target of failover (bystander standby databases) are disabled depends upon how much redo data they have applied relative to the failover target and the standby type of the failover target:

- If the failover target is a physical or snapshot standby database, the original primary database must be reinstated or re-created in order to be a standby database for the new primary database. In addition, some standby databases may be disabled by the broker during the failover if the broker detects that they have applied redo beyond where the new primary database had applied. Any standby database that was disabled by the broker must be reinstated or re-created, as described in [Reenabling Disabled Databases After a Role Change](#), before it can be a standby database for the new primary database.

Note that if failover was performed on a snapshot standby database, the old primary must be either reinstated or re-created as a physical standby database.

- If the failover target is a logical standby database, the original primary database and all physical and snapshot standby databases in the configuration will be disabled. The primary database can be reinstated if it had flashback database enabled. The physical and snapshot standby databases will have to be re-created from a copy of the new primary database. See [Reenabling Disabled Databases After a Role Change](#) for more information.

If the primary database can be mounted, it may be possible to flush any unsent redo data from the primary database to the target standby database using the `ALTER SYSTEM FLUSH REDO` SQL statement. If this operation is successful, a zero data loss failover may be possible even if the primary database is not in a zero data loss protection mode. See *Oracle Data Guard Concepts and Administration* for more information on using the `ALTER SYSTEM FLUSH REDO` statement.

During a complete failover, the broker performs the failover steps described in [How the Broker Performs a Complete Failover Operation](#).

- An *immediate* failover is the fastest type of failover. However, no additional data is applied on the standby database once you invoke the failover. Another consequence of immediate failover is that all other databases in the configuration are disabled and must be reinstated or re-created before they can serve as standby databases for the new primary database. [Reenabling Disabled Databases After a Role Change](#) describes how to do this. During an immediate failover, the broker performs the failover steps described in [How the Broker Performs an Immediate Failover Operation](#).

 **Note:**

Always try to perform a complete failover first unless redo apply has stopped at the failover target due to an ORA-752 or ORA-600 [3020] error. If one of these errors has occurred, follow the guidelines in "Resolving ORA-752 or ORA-600 [3020] During Standby Recovery" in My Oracle Support Note 1265884.1 before proceeding. This support note is available at <http://support.oracle.com>.

An immediate failover should only be performed when a complete failover is unsuccessful or in the error cases just noted. A complete failover can occur without any data loss, depending on the destination attributes of redo transport services, but an immediate failover usually results in some data loss.

Performing a Manual Failover Operation

Before beginning a failover, first determine that there is no possibility of recovering the primary database in a timely manner, and ensure that the primary database is shut down.

If the database is managed by Oracle Clusterware, broker does not open any pluggable databases (PDBs) on any of the instances. Instead, when broker notifies the Oracle Clusterware agent that the failover completed, the Oracle Clusterware agent opens PDBs on particular instances based on the service configuration.

The steps in this section describe the tasks involved to perform a manual failover. Depending on the failover and the types of standby databases involved, some of the databases may need to be reinstated or re-created.

- [Performing a Manual Failover Task 1: Determine Which of the Available Standby Databases is the Best Target for the Failover](#)
- [Performing a Manual Failover Task 2: Start the Failover](#)
- [Performing a Manual Failover Task 3: Reset the Protection Mode](#)
- [Performing a Manual Failover Task 4: Re-establish a Disaster-Recovery Configuration](#)

Performing a Manual Failover Task 1: Determine Which of the Available Standby Databases is the Best Target for the Failover

These are the guidelines for choosing a target standby database.

See [Choosing a Target Standby Database](#).

Performing a Manual Failover Task 2: Start the Failover

Use Cloud Control or DGMGRL to perform either a *complete* (recommended) or an *immediate* failover.

Manual Failover Using Cloud Control:

On the Oracle Data Guard Overview page in Cloud Control, select the standby database that you want to change to the primary role and click **Failover**. Then, on the Failover Confirmation page, click **Yes** to invoke the default Complete failover option.

Manual Failover Using DGMGRL:

Connect to the target standby database and issue the `FAILOVER` command to perform a failover, specifying the name of the standby database that you want to become the primary database:

```
DGMGRL> FAILOVER TO database-name;
```

Specify the optional `IMMEDIATE` clause to perform an immediate failover if any of the following conditions are true:

- An `ORA-752` error has occurred at the standby database
- An `ORA-600 [3020]` error has occurred at the standby database and Oracle support has determined that it was caused by a lost write at the primary database
- A complete failover is not possible

```
DGMGRL> FAILOVER TO database-name IMMEDIATE;
```

See Also:

- [FAILOVER](#)

If you are performing a complete failover, then all accumulated redo data is applied before the database role is changed to primary. If you are performing an immediate failover, then the database role is changed to primary without applying any accumulated redo data.

If the target is a snapshot standby database, the broker first converts the database to a physical standby database.

No instances are shutdown when doing a failover, if the target standby database is either a physical or logical standby. If the target standby database is a snapshot standby database, all of its instances must be restarted to the mount mode before performing failover. Both Cloud Control and the DGMGRL CLI will do this automatically as part of failover.

Performing a Manual Failover Task 3: Reset the Protection Mode

This list describes how the overall Oracle Data Guard protection mode is handled after a manual failover (complete or immediate).

- If the protection mode was at maximum protection, it is reset to maximum performance. You can upgrade the protection mode later, if necessary, as described in [Setting the Protection Mode for Your Configuration](#).
- If the protection mode was at maximum availability or maximum performance, it remains unchanged.

 **Note:**

If you perform a manual failover when fast-start failover is enabled:

- The failover can only be performed to the current target standby database.
- The broker preserves the protection mode that was in effect prior to the failover.

Performing a Manual Failover Task 4: Re-establish a Disaster-Recovery Configuration

To maintain a viable disaster-recovery solution in the event of another disaster, you may need to perform additional steps.

See [Reenabling Disabled Databases After a Role Change](#)

- Reinstatement the original primary database to act as a standby database in the new configuration.

 **Caution:**

Do not attempt to reinstate the old primary database if an `ORA-752` or `ORA-600 [3020]` error has occurred at the failover target. Instead, the old primary database must be re-created as a standby from a backup of the new primary using the procedure described in [How to Re-create and Reenable a Disabled Database](#).

- Reinstatement or re-creation of standby databases in the configuration that were disabled by the broker.

After a complete failover finishes, any bystander standby database that is not viable as a standby for the new primary database will be disabled by the broker. This can happen for either of the following reasons:

- A bystander standby database has applied more redo data than the new primary database itself had applied when it was a standby database. The standby database must be re-created or reinstated before it can serve as a standby for the new primary database.
- The failover was to a logical standby database. The broker disables all of the physical and snapshot standby databases in the configuration. They must be re-created before they can serve as standby to the new primary database.

How the Broker Performs a Complete Failover Operation

These are the actions the broker performs after you start a complete failover.

1. Verifies that the target standby database is enabled. If the database is not enabled, you will not be able to perform a failover to this database.

2. Waits for the target standby database to finish applying any unapplied redo data before stopping Redo Apply (if the target is a physical standby database) or SQL Apply (if the target is a logical standby database).

If the target is a snapshot standby database, the broker first converts the database back to a physical standby and then starts Redo Apply to apply all the accumulated redo before completing the failover and opening the database as a primary database.

3. Transitions the target standby database into the primary database role, as follows:
 - a. Changes the role of the database from standby to primary.
 - b. Opens the new primary database in read/write mode.
 - c. Determines whether or not any standby databases that did not participate in the failover operation have applied redo data beyond the new primary database, and thus need to be disabled.

If a bystander standby database is not disabled by the broker during this failover, it will remain in the state it was in before the failover. For example, if a physical standby database was in the `APPLY-OFF` state, it will remain in the `APPLY-OFF` state.

By default, the broker always determines whether bystander standby databases will be viable standby databases for the new primary when performing a complete failover. If you want the broker to skip this viability check of bystander standby databases during a complete failover, thus decreasing the overall failover time, set the `BystandersFollowRoleChange` configuration property to `NONE`.

When this property is set to `NONE`, the broker will disable all bystander standby databases without checking whether they have applied more redo data than the new primary database. You will have to reinstate or re-create (see [Reenabling Disabled Databases After a Role Change](#)) the standby databases after failover has completed. The `SHOW CONFIGURATION` command will show you which databases can be reinstated and which databases must be re-created. Use the `SHOW CONFIGURATION BystandersFollowRoleChange` command to see the value of this property. The default value is `ALL`.

This property also affects whether the broker skips viability checks of bystander standby databases when a fast-start failover occurs.

- d. Starts redo transport services to begin transmitting redo data to all bystander standby databases that were not disabled.

 **Note:**

Bystander standby databases may be disabled by the broker during the failover, and they must be reinstated or re-created before they can serve as standby databases to the new primary database. Oracle recommends configuring Flashback Database on every database so that if failover occurs to a physical standby database, you can more easily reinstate any disabled standby databases. If failover occurs to a logical standby database, all physical and snapshot standby databases will be disabled by the broker. In this case, Flashback Database cannot be used to reinstate databases. They must be re-created from a copy of the new primary database. Logical standby databases that are disabled during failover can be reinstated.

4. If the failover target database is an Oracle RAC physical or snapshot standby database, the broker directs Oracle Clusterware to restart all instances that may have been shut down prior to the failover.

The broker allows the failover to proceed as long as there are no errors for the standby database that you selected to participate in the failover. Errors occurring for any other configuration members will not impede the switchover. If you initiated a complete failover and it fails, you might need to use immediate failover.

Complete Failovers in Configurations Using Far Sync Instances

It is possible to manually perform a complete failover to a standby database that receives redo data from a far sync instance. To failover, connect to the standby database and use the DGMGRL `FAILOVER TO db-unique-name` command. Any unsent redo data residing on the far sync instance is transmitted to the target physical standby prior to converting the physical standby into a primary database.

Complete Failovers in Configurations Using Cascaded Standbys

In a complete failover, it is also possible to failover to a standby database (terminal standby) that gets redo from another standby database (cascader). In such a case, no attempt is made to transmit any unsent redo from the cascader to the terminal standby.

How the Broker Performs an Immediate Failover Operation

To start an immediate failover, use the DGMGRL `FAILOVER TO database-name IMMEDIATE` command.

Once an immediate failover is started, the broker:

1. Verifies that the target standby database is enabled. If the standby database is not enabled for management by the broker, then the failover cannot occur.
2. Stops Redo Apply or SQL Apply on the standby database immediately, without waiting until all available redo data has been applied. This may result in data loss.
3. Transitions the target standby database into the primary role, opens the new primary database in read/write mode, and starts redo transport services.

After an immediate failover completes, all the standby databases in the configuration, regardless of their type, are disabled. They may be reinstated if Flashback Database is enabled on those databases. Otherwise, they must be re-created from a copy of the new primary database.

The broker allows a complete failover to proceed as long as there are no errors present on the standby database that you selected to participate in the failover.

The broker allows an immediate failover to proceed even if there are errors present on the standby database that you selected to participate in the failover.

Immediate Failovers in Configurations Using Far Sync Instances

It is possible to manually perform an immediate failover to a standby database that receives redo data from a far sync instance. In this case, no attempt is made to transmit any unsent redo from the far sync instance to the target physical standby prior to converting the physical standby into a primary database.

Immediate Failovers in Configurations Using Cascaded Standbys

In an immediate failover, it is also possible to failover to a standby database (terminal standby) that gets redo from another standby database (cascader). In such a case, no attempt is made to transmit any unsent redo from the cascader to the terminal standby.

Reenabling Disabled Databases After a Role Change

To restore your original disaster-recovery solution after switchover to a logical standby database or after failover to any standby database, you may need to perform additional steps.

Databases that have been disabled after a role transition are not *removed* from the broker configuration, but they are no longer managed by the broker.

To reenabling broker management of these databases, you must reinstate or re-create the databases using one of the following procedures:

- If a database can be reinstated, the database will show the following status:

```
ORA-16661: the standby database needs to be reinstated
```

Reinstate the database using the DGMGRL `REINSTATE DATABASE` command or the `reinst` option in Cloud Control, as described in [How to Reinstate a Database](#). The broker automatically reenables the database as part of reinstating it.

- If a database must be re-created from a copy of the new primary database, it will have the following status:

```
ORA-16795: the standby database needs to be re-created
```

Re-create the standby database from a copy of the primary database and then reenabling it, as described in [How to Re-create and Reenable a Disabled Database](#).

Note:

Any database that was disabled while multiple role changes were performed cannot be reinstated. You must re-create the database manually from a copy of the current primary database and then reenabling the database in the broker configuration.

Whether you reinstate or re-create a database depends on whether you performed a switchover or failover, on the type of standby database that was the target of the operation, and on whether or not there are sufficient flashback logs. Note that role changes to logical standby databases always result in physical standby database bystanders being disabled. They cannot be reinstated. They must be re-created from a copy of the new primary database.

The following sections describe how to reinstate or reenabling a database.

How to Reinstate a Database

You can use the broker's `reinst` capability to make a failed primary database a viable standby database for the new primary.

This can be done regardless of whether the failover was done to a physical, logical, or snapshot standby database.

You can also reinstate bystander standby databases that were disabled during a failover operation.

Databases that can be reinstated will have the following status value:

```
ORA-16661: the standby database needs to be reinstated
```

If the database is managed by Oracle Clusterware, broker does not open any pluggable databases (PDBs) on any of the instances. Instead, when broker notifies the Oracle Clusterware agent that the failover completed, the Oracle Clusterware agent opens PDBs on particular instances based on the service configuration.

For the `REINSTATE` command to succeed, Flashback Database must have been enabled on the database prior to the failover and there must be sufficient flashback logs on that database. In addition, the database to be reinstated and the new primary database must have network connectivity.

To reinstate a database:

1. Restart the database to the mounted state
2. Connect to the new primary database
3. Use Cloud Control or DGMGRL to reinstate the database

The broker reinstates a failed primary database as a standby database of the same type (physical or logical standby database) as the old standby database. The only exception to this is failovers to snapshot standby databases. In such cases, the failed primary database is reinstated as a physical standby database.

The broker reinstates bystander standby databases that were disabled during a failover as standby databases to the new primary database.

Reinstatement Using Cloud Control

On the Oracle Data Guard Overview page, click **Database must be reinstated**. This brings up the General Properties page that provides a **Reinstate** button. After you click the **Reinstate** button, Cloud Control begins reinstating the database.

When the process is complete, the database will be enabled as a standby database to the new primary database, and Cloud Control displays the Oracle Data Guard Overview page.

Reinstatement Using DGMGRL

Issue the following command while connected to any database in the broker configuration, except the database that is to be reinstated:

```
DGMGRL> REINSTATE DATABASE db_unique_name;
```

The newly reinstated standby database will begin serving as a standby database to the new primary database. If reinstatement of a database fails, its status changes to `ORA-16795: the standby database needs to be re-created`. You must then re-create it from a copy of the new primary database and reenable it as described in [How to Re-create and Reenable a Disabled Database](#).

How to Re-create and Reenable a Disabled Database

If you re-create the old primary database, it must be created as the standby type of the old standby database.

For example, if the old standby was a physical or snapshot standby, then the old primary must be re-created as a physical standby.

After the database has been re-created, enable broker management of the re-created standby database by using the `DGMGRL ENABLE DATABASE` command.

If you performed a failover or switchover that requires you to re-create the failed primary database or standby databases that were disabled during the role transition, then follow the procedures in the *Oracle Data Guard Concepts and Administration* chapter, "Creating a Physical Standby Database" and also the *Oracle Data Guard Concepts and Administration* chapter, "Creating a Logical Standby Database."

Fast-Start Failover

Fast-start failover allows the broker to automatically fail over to a previously chosen standby database in the event of loss of the primary database.

The following sections describe these topics:

- [About Fast-start Failover](#)
- [Prerequisites for Enabling Fast-Start Failover](#)
- [Enabling Fast-Start Failover](#)
- [Viewing Fast-Start Failover Configuration Statistics and Status](#)
- [Disabling Fast-Start Failover](#)
- [Performance Considerations for Fast-Start Failover](#)
- [Managing the Observer](#)
- [Reinstating the Former Primary Database in the Broker Configuration](#)
- [Shutting Down Databases In a Fast-Start Failover Environment](#)

About Fast-start Failover

Fast-start failover quickly and reliably fails over the target standby database to the primary database role, without requiring you to perform any manual steps to invoke the failover.

If the primary database has multiple standby databases, then you can specify multiple fast-start failover targets, using the `FastStartFailoverTarget` property. The targets are referred to as candidate targets. The broker selects a target based on the order in which they are specified on the `FaststartFailoverTarget` property. If the designated fast-start failover target develops a problem and cannot be the target of a failover, then the broker automatically changes the fast-start failover target to one of the other candidate targets.

Fast-start failover can be used only in a broker configuration and can be configured only through DGMGRL or Cloud Control. The `VALIDATE FAST_START FAILOVER` command can be used to validate a fast-start failover configuration and identify misconfigurations that prevent the initiation of fast-start failover.

**See Also:**[VALIDATE FAST_START FAILOVER](#)

Fast-start Failover Protection Modes

You can use the maximum protection, maximum availability, or maximum performance protection mode with fast-start failover.

- **Maximum protection mode**
Provides an automatic failover environment guaranteed to lose no data.
- **Maximum availability mode**
Provides an automatic failover environment that is guaranteed to either lose no data (when the `FastStartFailoverLagLimit` configuration property is set to zero) or lose no more than the amount of data (in seconds) specified by the `FastStartFailoverLagLimit` configuration property.
- **Maximum performance mode**
Provides an automatic failover environment that is guaranteed to lose no more than the amount of data (in seconds) specified by the `FastStartFailoverLagLimit` configuration property.

Once fast-start failover is enabled, the broker will ensure that fast-start failover is only possible when the configured data loss guarantee can be upheld. If the configured data loss guarantee cannot be upheld, redo generation on the primary database will be stalled. In maximum availability and maximum performance modes, to avoid a prolonged stall, either the observer or target standby database may allow the primary database to continue redo generation after first recording that a fast-start failover cannot happen.

In maximum protection mode, an automatic failover is always possible because the broker does not allow the primary database to commit transactions until it has regained connectivity with target standby. Therefore, the target standby never falls behind the primary (as it might in maximum availability and maximum performance modes).

In maximum availability mode, the behavior depends on the value of the `FastStartFailoverLagLimit` property. If the value is zero, the standby must have received all the redo data the primary has generated in order for automatic failover to occur. If the value is non-zero, failover is possible any time the standby database's apply lag is less than or equal to the value specified by the `FastStartFailoverLagLimit` property.

In maximum performance mode, the ability to automatically failover is restored once the target standby database's redo applied point is no longer lagging behind the primary database's redo generation point by more than the value specified by the `FastStartFailoverLagLimit` property.

The following table summarizes which standby types are supported in which protection modes when fast-start failover is enabled. (Snapshot standbys are not included in the table because they are not supported as fast-start failover targets.)

Protection Mode	Physical Standbys Supported?	Logical Standbys Supported?	Far Sync Instances Supported to Send Redo?
Maximum Protection	Yes	No	No
Maximum Availability	Yes	Yes	Yes
Maximum Performance	Yes	Yes	Yes

As shown in the table, fast-start failover can be enabled in maximum availability mode when the fast-start failover target is a logical or physical standby database that receives redo data from a far sync instance. This lets you take advantage of the broker's automatic failover feature in configurations set up for zero data loss protection at any distance.

Related Topics

- [Scenario 7: Enabling Fast-Start Failover When a Far Sync Instance Is In Use](#)
Fast-start failover can be enabled in maximum availability mode when the fast-start failover target is a logical or physical standby database that receives redo data from a far sync instance.

Observers in a Fast-start Failover Configuration

An observer is a separate OCI client-side component that runs on a different computer from the primary and standby databases and monitors the availability of the primary database.

Observer sites monitor the fast-start failover environment. Once an observer is started, no further user interaction is required. If both the observer and designated standby database lose connectivity with the primary database for longer than the number of seconds specified by the `FastStartFailoverThreshold` configuration property, the observer will initiate a fast-start failover to the standby database. In addition, the primary database will shut down if it perceives a loss of connectivity for a period longer than `FastStartFailoverThreshold` seconds, if the `FastStartFailoverPmyShutdown` configuration property is set to `TRUE`. After the failover completes, the former primary database is automatically reinstated as a standby database when a connection to it is reestablished, if the `FastStartFailoverAutoReinstate` configuration property is set to `TRUE`.

Note:

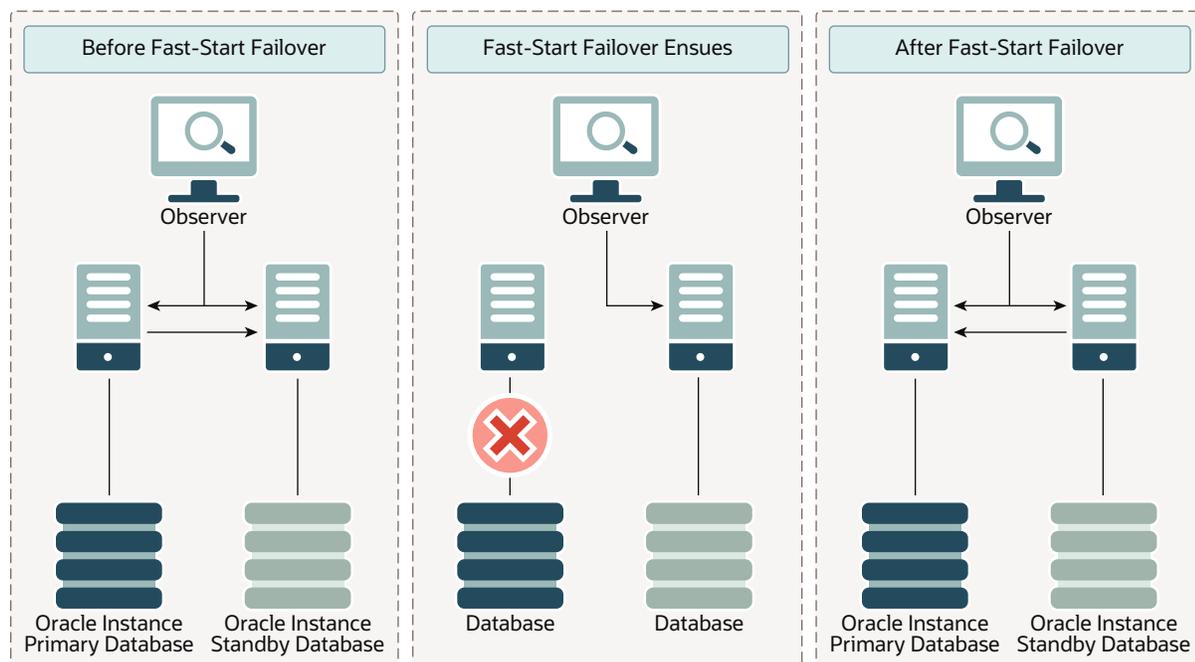
When a fast-start failover occurs because either a user configurable fast-start failover condition is detected or an application initiates a fast-start failover by calling the `DBMS_DG.INITIATE_FS_FAILOVER` function, the former primary database is always shut down and never automatically reinstated. This is true regardless of the settings for the `FastStartFailoverPmyShutdown` and `FastStartFailoverAutoReinstate` configuration properties. See [Enabling Fast-Start Failover](#) for more information.

Relationship Between Primary, Target Standby, and Observer During Fast-start Failover

Figure 6-1 shows the relationships between the primary database, target standby database, and observer during fast-start failover:

- Before Fast-Start Failover:** Oracle Data Guard is operating in a steady state, with the primary database transmitting redo data to the target standby database and the observer monitoring the state of the entire configuration.
- FastStart Failover Ensues:** Disaster strikes the primary database and its network connections to both the observer and the target standby database are lost. Upon detecting the break in communication, the observer attempts to reestablish a connection with the primary database for the amount of time defined by the `FastStartFailoverThreshold` property before initiating a fast-start failover. If the observer is unable to regain a connection to the primary database within the specified time, and the target standby database is ready for fast-start failover, then fast-start failover ensues.
- After Fast-Start Failover:** The fast-start failover has completed and the target standby database is running in the primary database role. After the former primary database has been repaired, the observer reestablishes its connection to that database and reinstates it as a new standby database. The new primary database starts transmitting redo data to the new standby database.

Figure 6-1 Relationship of Primary and Standby Databases and the Observer



See Also:

[Managing the Observer](#) for information about configuring and managing observers

Location of Client-side Broker Files

Client-side broker files include the observer configuration file (`observer.ora`), observer log file, observer runtime data file (`fsfo.dat`), fast-start failover callout configuration file, and fast-start failover callout script files.

Starting with Oracle Database Release 21c, use the `DG_ADMIN` environment variable to specify the default location for client-side broker files. These files are stored in subdirectories of the `DG_ADMIN` directory. You must create the directory specified by the `DG_ADMIN` environment variable and ensure that it has the required permissions.

If the `DG_ADMIN` environment variable is not set, or the directory by this environment variable does not exist, or the `$DG_ADMIN` directory does not have the required permissions, broker does the following:

- Stores the observer runtime data file and observer configuration file in the current working directory
- Uses standard output for displaying the observer logs



Note:

Because callout configuration and script files must be stored in a subdirectory of the `$DG_ADMIN` directory, fast-start failover callout functionality will not work if the `DG_ADMIN` environment variable is not set correctly.

When you run DGMGRL commands, if a path and file name are explicitly specified for client-side broker files, the specified values are used. If only a file name is specified, the file is stored in an appropriate directory under the broker's `$DG_ADMIN` directory. If only a path is specified, the files are stored in the specified path using the default file names. If the `DG_ADMIN` environment variable is not set, the files are stored in the current working directory.

Table 6-1 Content of Default Directory for Client-side Files

Directory Name	Description
admin	<p>Contains the observer configuration file that is used by DGMGRL to manage multiple observers on multiple configurations. This file also declares broker configurations and defines configuration groups used by multiple configuration commands. The default name for the observer configuration file is <code>observer.ora</code>.</p> <p>When DGMGRL starts, if the <code>DG_ADMIN</code> environment variable is set and the specified directory has the required permissions, the <code>admin</code> folder is created under the <code>\$DG_ADMIN</code> directory.</p> <p>When you run commands that need access to the observer configuration file, such as <code>START OBSERVING</code>, <code>STOP OBSERVING</code>, and <code>SET MASTEROBSERVERHOSTS</code>, DGMGRL reports an error if the directory does not have the required permissions.</p>

Table 6-1 (Cont.) Content of Default Directory for Client-side Files

Directory Name	Description
<code>config_ConfigurationSimpleName</code>	<p>Stores files related to the observer and callout configuration. This directory has the same permissions as its parent directory.</p> <p>For each broker configuration on which one or more observers are registered, a directory named <code>ConfigurationSimpleName</code> is created. <code>ConfigurationSimpleName</code> represents an alias of the broker configuration name. Subdirectories within this directory are used to store the files related to the configuration.</p> <p>This directory is created when you run the <code>CONNECT</code> command. When running the <code>START OBSERVER</code> command, if this directory does not have the required permissions, DGMGRL reports an error.</p>
<code>config_ConfigurationSimpleName/log</code>	<p>Contains the observer log file for the broker configuration named <code>ConfigurationSimpleName</code>. The default name of the observer log file is <code>observer_hostname.log</code>.</p>
<code>config_ConfigurationSimpleName/dat</code>	<p>Contains the observer runtime data file for the broker configuration named <code>ConfigurationSimpleName</code>. The default name of the observer runtime data file is <code>fsfo_hostname.dat</code>. This file contains important information about the observer. In the event of a crash, data in this file can be used to restart the observer to the status before the crash.</p>
<code>config_ConfigurationSimpleName/callout</code>	<p>Contains the callout configuration file, pre-callout script, post-callout script, and pre-callout success file for the broker configuration named <code>ConfigurationSimpleName</code>. The default name of the callout configuration file is <code>fsfocallout.ora</code>.</p>

**Note:**

If the database uses a version lower than Oracle Database Release 21c, the `ConfigurationSimpleName` is derived from the `DB_UNIQUE_NAME` of the database. See [ConfigurationSimpleName](#).

Permissions Required by the `DG_ADMIN` Directory

Ensure that the required permissions are granted to the `DG_ADMIN` directory.

On Linux/Unix, the directory specified by the `DG_ADMIN` environment variable must have read, write, and execute permissions for the directory owner only. The subdirectories that DGMGRL creates under this directory will also have the same permissions.

On Windows, the directory specified by the `DG_ADMIN` environment variable must have exclusive permissions wherein it can be accessed only by the current operating system user who is running DGMGRL. The subdirectories created under this directory by DGMGRL will also have the same permissions.

Fast-start Failover Callout Configuration Files

Use the callout configuration file and script files to automate tasks that must be performed before and after a fast-start failover operation.

The name of the callout configuration file is `fsfocallout.ora`. You cannot use a different name for this file. This file is stored in the `$DG_ADMIN/config_ConfigurationSimpleName/callout` directory. If the `DG_ADMIN` environment variable is not defined, or the directory specified by this variable does not exist, or the directory does not have the required permissions, fast-start failover callouts will fail.

The name of the callout configuration scripts is specified in `fsfocallout.ora`. These scripts must be in the same directory as the callout configuration file. You can create two callout configuration scripts, a pre-callout configuration script and post-callout configuration script. Before a fast-start failover operation, the observer checks if a fast-start failover configuration file exists. If it exists, and it contains a pre-callout script location, this script is run before the fast-start failover is initiated. The pre-callout script must create a `.suc` and `.err` file in the `callout` directory. The existence of a `.suc` file, irrespective of its content, indicates that the script executed successfully. After fast-start failover succeeds, if a post-callout script is specified in the fast-start failover configuration file, this script is run.

The `VALIDATE FAST_START FAILOVER` command parses the callout configuration scripts and checks for errors or misconfigurations.



Note:

[Enabling Fast-Start Failover Task 7: Configure Actions Before and After Fast-start Failover \(Optional\)](#)

Prerequisites for Enabling Fast-Start Failover

There are prerequisites that must be met before the broker allows you to enable fast-start failover.

The prerequisites are as follows:

- Configure the protection mode. See [Setting the Protection Mode for Your Configuration](#).
- The selected standby database that will be the fast-start failover target must receive redo directly from the primary database.
- Ensure that the standby database you choose to be the target of fast-start failover has its `LogXptMode` property set as follows, depending on the protection mode:
 - Maximum protection — `SYNC`
 - Maximum availability — `SYNC` or `FASTSYNC` or `ASYNC`
 - Maximum performance — `SYNC` or `FASTSYNC` or `ASYNC`

The current primary database must have its `LogXptMode` property set accordingly and must have standby redo logs configured. Alternatively, use the `RedoRoutes` property to configure the redo transport mode for the target standby and the database currently in the primary role.

- To use a far sync instance with fast-start failover, the far sync instance transport mode must be set to either `SYNC` or `FASTSYNC` and the target standby database transport mode must be set to `ASYNC`. A far-sync instance cannot be used in maximum protection mode.
- Install the DGMGRL command-line interface on the observer computer as described in [Oracle Data Guard Installation](#).
- Configure the `TNSNAMES.ORA` file on the observer system so that the observer is able to connect to the primary database and to the pre-selected target standby database.
- If you are not using Oracle Clusterware or Oracle Restart, then you must create static service names so that the observer can automatically restart a database as part of reinstatement. Use the `VALIDATE STATIC CONNECT IDENTIFIER` command to ensure the static services have been configured correctly. See [Prerequisites](#) for more information.

Enabling Fast-Start Failover

You can enable fast-start failover from any site while connected to any database in the broker configuration.

Enabling fast-start failover does not trigger a failover. Instead, it allows an observer that is monitoring the configuration to initiate a fast-start failover should database conditions warrant a failover. (If there are other conditions, unique to an application, that would warrant a fast-start failover then the application can be set up to call the `DBMS_DG.INITIATE_FS_FAILOVER` function and start a fast-start failover immediately should any of those conditions occur. See [Directing a Fast-Start Failover From an Application](#))

Enabling fast-start failover and starting an observer process involves the following tasks. These tasks assume that you are connected as `SYS` or `SYSDG` and that a primary and standby database are already set up in a broker configuration.

- [Enabling Fast-Start Failover Task 1: Determine Which of the Available Standby Databases is the Best Target for the Failover](#)
- [Enabling Fast-Start Failover Task 2: Specify Target Standbys with the FastStartFailoverTarget Configuration Property](#)
- [Enabling Fast-Start Failover Task 3: Determine the Protection Mode You Want](#)
- [Enabling Fast-Start Failover Task 4: Set the FastStartFailoverThreshold Configuration Property](#)
- [Enabling Fast-Start Failover Task 5: Set Other Properties Related to Fast-Start Failover \(Optional\)](#)
- [Enabling Fast-Start Failover Task 6: Enable Additional Fast-Start Failover Conditions \(Optional\)](#)
- [Enabling Fast-Start Failover Task 7: Using DGMGRL or Cloud Control](#)
- [Enabling Fast-Start Failover Task 8: Start the Observer](#)

- [Enabling Fast-Start Failover Task 9: Verify the Fast-Start Failover Environment](#)

Enabling Fast-Start Failover Task 1: Determine Which Available Standby Databases Should Be Targets for the Failover

You must determine which available standby databases should be targets for failover.

Follow the guidelines described in [Choosing a Target Standby Database](#).

Enabling Fast-Start Failover Task 2: Specify Target Standby Databases with the FastStartFailoverTarget Configuration Property

Use the `FastStartFailoverTarget` configuration property on the current primary database to specify one or more fast-start failover targets.

The broker selects a target standby based on the order they are specified in the property.

- If there is only one standby database in the configuration, you can skip this step and continue with Task 3.
- If there is more than one standby database in the configuration, you must explicitly set the `FastStartFailoverTarget` property on the primary database to name one or more candidate target standby databases. When enabling fast-start failover, the broker verifies that the property indicates an existing standby. (Note that the target standby cannot be a far-sync instance. However the target can receive redo from a far sync instance.)

Note:

To change the `FastStartFailoverTarget` property to point to a different standby database, disable fast-start failover, set the `FastStartFailoverTarget` property, and reenables fast-start failover.

See [FastStartFailoverTarget](#) for more information about this property.

Enabling Fast-Start Failover Task 3: Determine the Protection Mode You Want

If you cannot tolerate any loss of data, then ensure that the configuration protection mode is set to maximum availability or maximum protection. For zero data loss in maximum availability mode, the `FastStartFailoverLagLimit` property must be set to zero.

In maximum availability mode, set the `LogXptMode` database property for both the primary and target standby databases to `SYNC` or `FASTSYNC`. In maximum protection mode, set the `LogXptMode` database property to `SYNC` (note that in maximum protection mode, a far sync instance cannot be used to ship redo to a standby). The following is an example of setting the `LogXptMode` property:

```
DGMGRL> EDIT DATABASE 'North_Sales' SET PROPERTY LogXptMode=SYNC;
DGMGRL> EDIT DATABASE 'South_Sales' SET PROPERTY LogXptMode=SYNC;
DGMGRL> EDIT CONFIGURATION SET PROTECTION MODE AS MaxAvailability;
```

Alternatively, use the `RedoRoutes` property to set the redo transport mode for the target standby and database that is currently in the primary role. Then set the configuration protection mode to maximum availability.

If you are more concerned about the performance of the primary database than a minimal loss of data, consider enabling fast-start failover when the configuration protection mode is set to maximum performance. In this mode you will need to consider how much data loss is acceptable in terms of seconds and set the `FastStartFailoverLagLimit` configuration property accordingly. This property specifies the amount of data, in seconds, that the target standby database can lag behind the primary database in terms of redo applied. If the standby database's redo applied point is within that many seconds of the primary database's redo generation point, a fast-start failover will be allowed. The `FastStartFailoverLagLimit` configuration property is only used by the broker when enabling fast-start failover for configurations operating in maximum performance mode. The default value is 30 seconds and the lowest possible value is 5 seconds.

In addition to setting the configuration protection mode to maximum performance, you will also need to ensure that the `LogXptMode` database property for both the primary and target standby database is set to `ASync`. For example:

```
DGMGRL> EDIT DATABASE 'North_Sales' SET PROPERTY LogXptMode=ASync;  
DGMGRL> EDIT DATABASE 'South_Sales' SET PROPERTY LogXptMode=ASync;  
DGMGRL> EDIT CONFIGURATION SET PROTECTION MODE AS MaxPerformance;  
DGMGRL> EDIT CONFIGURATION SET PROPERTY FastStartFailoverLagLimit=45;
```

Enabling Fast-Start Failover Task 4: Set the `FastStartFailoverThreshold` Configuration Property

Fast-start failover occurs if both the observer and the target standby database lose connection to the primary database for the period of time specified by the `FastStartFailoverThreshold` configuration property.

Set the `FastStartFailoverThreshold` property to specify the number of seconds you want the observer and target standby database to wait (after detecting the primary database is unavailable) before initiating a failover. For example:

```
DGMGRL> EDIT CONFIGURATION SET PROPERTY FastStartFailoverThreshold = 45;
```

The default value for the `FastStartFailoverThreshold` property is 30 seconds and the lowest possible value is 6 seconds. If you have an Oracle RAC primary database, consider specifying a higher value to minimize the possibility of a false failover in the event of an instance failure.

The time interval starts when the observer first loses its connection to the primary database. If the observer is unable to regain a connection to the primary database within the specified time, then the observer begins a fast-start failover provided the standby database is ready to fail over. Although the default value of 30 seconds is typically adequate for detecting outages and failures on most configurations, you can adjust failover sensitivity with this property to decrease the probability of false failovers in a temporarily unstable environment.

If the `FastStartFailoverPmyShutdown` configuration property is set to `TRUE`, the primary database will shut down after `FastStartFailoverThreshold` seconds has elapsed if redo generation has been stalled and the primary database is unable to reestablish connectivity with either the observer or target standby database.

Note that the `FastStartFailoverThreshold` property can be changed even when fast-start failover is enabled.

**See Also:**

[FastStartFailoverThreshold](#) for reference information about the `FastStartFailoverThreshold` property

Enabling Fast-Start Failover Task 5: Set Other Properties Related to Fast-Start Failover (Optional)

This table describes the optional database properties that you can set.

Property Name	Description	Default Value
<code>FastStartFailoverPmyShutdown</code>	This configuration property causes the primary database to shut down if fast-start failover is enabled and <code>V\$DATABASE.FS_FAILOVER_STATUS</code> indicates the primary has been <code>STALLED</code> for longer than <code>FastStartFailoverThreshold</code> seconds. A value of <code>TRUE</code> helps to ensure that an isolated primary database cannot satisfy user queries. This property cannot be used to prevent the primary database from shutting down if a fast-start failover occurred because a user configuration condition was detected or was requested by an application by calling the <code>DBMS_DG.INITIATE_FS_FAILOVER</code> function.	<code>TRUE</code>
<code>FastStartFailoverLagLimit</code>	This configuration property establishes an acceptable limit, in seconds, that the standby is allowed to fall behind the primary in terms of redo applied, beyond which a fast-start failover will not be allowed. The lowest possible value is 5 seconds.	30 seconds
<code>FastStartFailoverAutoReinstat e</code>	This configuration property causes the former primary database to be automatically reinstated if a fast-start failover was initiated because the primary database was either isolated or had crashed. To prevent automatic reinstatement of the former primary database in these cases, set this configuration property to <code>FALSE</code> . The broker never automatically reinstates the former primary database if a fast-start failover was initiated because a user configuration condition was detected or was requested by an application calling the <code>DBMS_DG.INITIATE_FS_FAILOVER</code> function.	<code>TRUE</code>

Property Name	Description	Default Value
ObserverConnectIdentifier	This database property is used to specify how the observer should connect to and monitor the primary and standby database. Set this property for the primary and target standby database if you want the observer to use a different connect identifier than that used to ship redo data (that is, the connect identifier specified by the <code>DGConnectIdentifier</code> property).	Observer uses the value of the <code>DGConnectIdentifier</code> property to connect to and monitor the primary and target standby databases.
ObserverOverride	The <code>ObserverOverride</code> configuration property, when set to <code>TRUE</code> , allows an automatic failover to occur when the observer has lost connectivity to the primary, even if the standby has a healthy connection to the primary.	<code>FALSE</code>
ObserverPingInterval	The <code>ObserverPingInterval</code> configuration property specifies how frequently the observer must ping the primary database. This property is measured in milliseconds. The minimum value is 100 milliseconds. To achieve lower detection times for primary database failures, you must set the <code>ObserverPingInterval</code> and <code>ObserverPingRetry</code> properties before enabling fast-start failover.	Valid values are ≥ 100 . Default value is 100 milliseconds.
ObserverPingRetry	The <code>ObserverPingRetry</code> property specifies the number of times that the observer retries a failed ping before it initiates a failover to the target standby database. A failed ping is a ping to the primary database that failed or took longer than the time specified by the <code>ObserverPingInterval</code> property. You must set both the <code>ObserverPingRetry</code> and <code>ObserverPingInterval</code> properties to achieve lower detection times for primary database failures. The minimum value is 10. Therefore, the detection time can be reduced to nearly 1 second.	Valid values are ≥ 10 . Default value is 10 milliseconds.

Property Name	Description	Default Value
ObserverReconnect	The <code>ObserverReconnect</code> configuration property specifies how often the observer establishes a new connection to the primary database. When this property is set to the default value of 0, it prevents the observer from periodically establishing a new connection with the primary database. While this eliminates the processing overhead associated with periodically establishing a new observer connection to the primary database, it also prevents the observer from detecting that it is not possible to create new connections to the primary database. Oracle recommends that this property be set to a value that is small enough to allow timely detection of faults at the primary database, but large enough to limit the overhead associated with periodic observer connections to an acceptable level.	0 (zero)

Enabling Fast-Start Failover Task 6: Enable Additional Fast-Start Failover Conditions (Optional)

By default, a fast-start failover is done when both the observer and the standby cannot reach the primary after the configured time threshold (`FastStartFailoverThreshold`) has passed.

You can optionally indicate the database health conditions that should cause fast-start failover to occur. These conditions are described in the following table:

Health Condition	Description	Enabled by Default
Datafile Write Errors	If fast-start failover is enabled and the Datafile Write Errors condition is specified, then a fast-start failover is initiated if write errors are encountered in any data files, including temp files, system data files, and undo files.	Yes
Corrupted Dictionary	Dictionary corruption of a critical database. Currently, this state can be detected only when the database is open	Yes
Corrupted Controlfile	Controlfile is permanently damaged because of a disk failure.	Yes
Inaccessible Logfile	LGWR is unable to write to any member of the log group because on an I/O error	No
Stuck Archiver	Archiver is unable to archive a redo log because the device is full or unavailable.	No

In Oracle RAC configurations, the Inaccessible Logfile and Stuck Archiver health conditions may only be applicable to a single instance. Careful consideration should be given before enabling fast-start failover for either of these conditions because doing so will supersede availability options provided by Oracle Clusterware.

You can specify particular conditions for which a fast-start failover should occur using either Cloud Control or the `DGMGRL ENABLE FAST_START_FAILOVER_CONDITION` and `DISABLE FAST_START_FAILOVER_CONDITION` commands.

Enabling Fast-Start Failover Task 7: Configure Actions Before and After Fast-start Failover (Optional)

Tasks that must be performed before and after a fast-start failover operation can be automated using callout scripts.

To perform specified actions before or after a fast-start failover operation:

1. Create a pre-callout script, or a post-callout script, or both.
See [Fast-start Failover Callout Configuration Files](#) for information about the directory used to store callout scripts and the permissions required.
2. Create or update the fast-start failover callout configuration file and include the names of the scripts created in the previous step.

Example 6-1 Fast-start Failover Configuration File

The following example displays the contents of the fast-start failover configuration file /home1/dataguard/config_NorthSales/callout/fsfocallout.ora. The callout configuration scripts `fsfo_precallout` and `fsfo_postcallout` are stored in the same location as `fsfocallout.ora` and they have the required permissions.

```
# This is a fast-start failover configuration file.

# The pre-callout script that is run before fast-start failover is
enabled.
FastStartFailoverPreCallout=fsfo_precallout

# The timeout value (in seconds) for pre-callout script
FastStartFailoverPreCalloutTimeout=1200

# The name of the suc file created by the pre-callout script.
FastStartFailoverPreCalloutSucFileName=fsfo_precallout.suc

# The name of the error file that the pre-callout script creates
FastStartFailoverPreCalloutErrorFileName=precallout.err

# Action taken by observer if the suc file does not exist after
FastStartFailoverPreCalloutTimeout seconds
# or if an error file is detected before
FastStartFailoverPreCalloutTimeout seconds passed
FastStartFailoverActionOnPreCalloutFailure=STOP

# The post-callout script that is run after fast-start failover
succeeds
FastStartFailoverPostCallout=fsfo_postcallout
```

Enabling Fast-Start Failover Task 8: Using DGMGRL or Cloud Control

Use the Cloud Control Fast-Start Failover wizard or the DGMGRL `ENABLE FAST_START FAILOVER` command to enable fast-start failover.

To enable fast-start failover, both the primary and target standby databases must be running and have connectivity, and satisfy all of the prerequisite conditions listed in [Prerequisites for Enabling Fast-Start Failover](#).

Note:

The Cloud Control Fast-Start Failover wizard does not currently support specification of multiple targets. If you want to specify multiple targets, then you must use the DGMGRL utility.

Enable Fast-Start Failover Using Cloud Control

To enable fast-start failover in Cloud Control, use the Fast-Start Failover wizard. On the Oracle Data Guard Overview page next to the Fast-Start Failover status field, click **Disabled** to invoke the Fast-Start Failover page. Then, on the Fast-Start Failover Change Mode page, click **Enabled**. Cloud Control will start the observer. Then, on the Fast-Start Failover Configure page, select the standby database that should be the target of a failover. See [Choosing a Target Standby Database](#) for helpful advice. This page will not allow you to alter the protection mode. Rather, fast-start failover will be enabled in accordance with the current protection mode. If the currently configured mode is maximum protection, Cloud Control will downgrade the mode to maximum availability.

Enable Fast-Start Failover Using DGMGRL

To enable fast-start failover with DGMGRL, issue the `ENABLE FAST_START FAILOVER` command while connected to any database in the broker configuration, including on the observer computer. For example:

```
DGMGRL> ENABLE FAST_START FAILOVER;  
Enabled.
```

See Also:

- [ENABLE FAST_START FAILOVER](#)
- [Scenario 6: Enabling Fast-Start Failover and Starting the Observer](#)

Note:

Administration at the target standby site should be as comprehensive as that at the primary site because the standby database may assume the primary role without prior notice. Staff support, hardware and software, security (both software and site), network connections, and bandwidth should be equivalent at both sites.

Enabling Fast-Start Failover Task 9: Start the Observer

The primary database must be running in order to start the observer.

You can start the observer before or after you enable fast-start failover. If fast-start failover is already enabled, the observer immediately begins monitoring the status and connections to the primary and target standby databases. If fast-start failover is not already enabled, the observer waits until fast-start failover gets enabled and then begins monitoring.

Starting the Observer Using Cloud Control

If the Cloud Control agent is installed on the observer computer, it automatically starts the observer when you enable fast-start failover through Cloud Control. If the agent is not present, you must start the observer manually using the following instructions for the DGMGRL command-line interface.

Starting the Observer Using DGMGRL

To start the observer with DGMGRL, issue the following command on the observer computer:

```
DGMGRL> START OBSERVER;
```

The observer is a continuously executing process that is created when the `START OBSERVER` command is issued. Thus, the command-line prompt on the observer computer does not return until you issue the `STOP OBSERVER` command from another DGMGRL session. To issue commands and interact with the broker configuration, you must connect through another DGMGRL client session.

There can be up to four observers for a single Data Guard configuration. One is the master observer and the others are backup observers.

Starting the Observer as a Background Process Using DGMGRL

To start an observer as a background process, use the DGMGRL command `START OBSERVER IN BACKGROUND`. If this command is submitted successfully, the command-line prompt on the observer computer is returned to you so that you can continue to issue commands and interact with the broker configuration. This is the preferred method for starting an observer.

- See [Installing and Starting the Observer](#).
- See [Managing the Observer](#)
- See the [START OBSERVER](#) command for more information about starting the observer as a foreground process.
- See [START OBSERVER IN BACKGROUND](#) for more information about starting the observer as a background process.

Specifying Preferred Observers Based on Current Primary

To specify which observer can be a master observer when a database is in the primary role, use the `PreferredObserverHosts` property. This property allows you to specify a priority ordered list of the observers that can become the master observer when that database is in the primary role.

For example:

```
EDIT DATABASE North_Sales SET PROPERTY PreferredObserverHosts="'ob1-host:1,
ob2-host:2';
EDIT DATABASE South_Sales SET PROPERTY PreferredObserverHosts="'ob3-host:1,
ob4-host:2';
```

In this case, only observers on `ob1-host` and `ob2-host` can be a master observer when `North_Sales` is in the primary role. The observer on `ob1-host` will be given priority over the observer on `ob2-host` to become the master observer. If both of those observers are unavailable, the observers on `ob3-host` and `ob4-host` will not become the master observer.

Enabling Fast-Start Failover Task 10: Verify the Fast-Start Failover Environment

To verify the readiness of the fast-start failover configuration, issue the `DGMGRL SHOW CONFIGURATION VERBOSE` command or the `SHOW FAST_START FAILOVER` command on the primary database.

For example:

```
DGMGRL> SHOW FAST_START FAILOVER;

Fast-Start Failover: Enabled in Zero Data Loss Mode

Protection Mode: MaxAvailability
Lag Limit: 0 seconds

Threshold: 180 seconds
Ping Interval: 3000 milliseconds
Ping Retry: 0
Active Target: South_Sales
Potential Targets: East_Sales, West_Sales
   East_Sales    valid
   West_Sales    valid
Observer: observer.example.com
Shutdown Primary: TRUE
Auto-reinstate: TRUE
Observer Reconnect: (none)
Observer Override: FALSE

Configurable Failover Conditions
Health Conditions:
  Corrupted Controlfile      YES
  Corrupted Dictionary      YES
  Inaccessible Logfile      NO
  Stuck Archiver            NO
  Datafile Write Errors     YES

Oracle Error Conditions:
(none)
```

The following sections provide more information about the fast-start failover environment:

- [When Fast-Start Failover Is Enabled and the Observer Is Running](#)
- [Restrictions When Fast-Start Failover is Enabled](#)
- [Shutting Down the Primary Database When Fast-Start Failover Is Enabled](#)

- [Performing Manual Role Changes When Fast-Start Failover Is Enabled](#)

When Fast-Start Failover Is Enabled and the Observer Is Running

After fast-start failover is enabled and up to four observers are started, one observer is nominated as the master observer that continuously monitors the environment to ensure the primary database is available.

This section lists the steps the master observer takes to determine if a fast-start failover is needed and then to perform one, if necessary.

1. Monitor the environment to ensure the primary database is available.

The master observer waits the number of seconds specified by the `FastStartFailoverThreshold` configuration property before attempting a fast-start failover when the primary database has crashed or has lost connectivity with the observer, as in the following situations:

- The primary database loses its connections with both the observer and target standby database

- Instance failures

If a single-instance primary database (either Oracle RAC or non-Oracle RAC), or if all instances of an Oracle RAC primary database fail, the observer attempts a fast-start failover.

- Shutdown abort

If a single-instance primary database (either Oracle RAC or non-Oracle RAC), or if all instances of an Oracle RAC primary database are shut down with the `ABORT` option, the observer attempts a fast-start failover. Fast-start failover will not be attempted for the other types of database shutdown (`NORMAL`, `IMMEDIATE`, `TRANSACTIONAL`).

The master observer never waits for the threshold to expire to perform a fast-start failover in the following situations:

- User-configurable condition

If the master observer determines that any of the user-configurable conditions has been detected, then it attempts a fast-start failover.

- Application calls to `DBMS_DG.INITIATE_FS_FAILOVER`

If an application has called this function and it has received a status of `SUCCESS`, then the master observer attempts a fast-start failover.

2. Reconnect within the time specified by the `FastStartFailoverThreshold` property.

If the master observer detects an availability problem with the primary database, then it typically attempts to reconnect to the primary database within the time specified by the `FastStartFailoverThreshold` configuration property. The `FastStartFailoverThreshold` time interval starts when the observer first detects there might be a failure with the primary database.

The time interval specified by the `FastStartFailoverThreshold` property is ignored if the master observer detects that a user-configurable condition has occurred or if a fast-start failover has been requested by the `DBMS_DG.INITIATE_FS_FAILOVER` function.

If the primary database is an Oracle Real Application Clusters (Oracle RAC) database, the master observer will attempt to connect to one of the remaining primary instances. Fast-start failover will not occur unless all instances comprising the Oracle RAC primary database are perceived to have failed. The master observer uses the value specified by either the `DGConnectIdentifier` or `ObserverConnectIdentifier` database properties to connect to the primary and fast-start failover target standby databases. The value specified for either of these properties should allow the master observer to connect to any instance of an Oracle RAC database.

3. Verify the target standby database is ready for failover.

If fast-start failover is initiated, the master observer verifies the target standby database is ready to fail over to the primary database role.

Fast-start failover *cannot* occur if:

- Fast-start failover is no longer enabled
- The master observer cannot connect to the target standby database



See Also:

[What Happens if the Observer Fails?](#) if the observer is not running

- The master observer and the target standby database are inconsistent with regard to the current state of the broker configuration
 - The master observer is not running
 - If the protection mode is maximum availability or maximum protection and the target standby database was not synchronized with the primary database at the time the primary database failed
 - If the protection mode is maximum performance and the apply point of the target standby database lags the redo generation point of the primary database by more than the amount specified by the `FastStartFailoverLagLimit` configuration property at the time the primary database failed
 - The target standby database has contact with the primary database. However, failover is attempted if the `ObserverOverride` configuration property is set to `TRUE`.
 - The `FS_FAILOVER_STATUS` column in the `V$DATABASE` view for the target standby database displays a reason why fast-start failover cannot occur
 - A manual failover is already in progress. See [Manual Failover](#) for complete information about manual failovers.
 - The primary database was shut down without using the `ABORT` option
4. Initiate a fast-start failover.

If the target standby database is ready for failover, then the master observer immediately directs the target standby database to fail over to the primary database role. If failover is not possible for some reason, then the master observer will continue checking whether the standby database is ready to fail over. But it will also continue trying to reconnect to the primary database indefinitely. If it reconnects to the primary database before the standby agrees to fail over, then the master observer will stop attempting to initiate a fast-start failover.

5. Reinststate the former primary database as a new standby database.

After the fast-start failover completes successfully, the master observer will attempt to reinstate the former primary database as a new standby database when a connection to the former primary database is reestablished, and the `FastStartFailoverAutoReinstate` configuration property is set to `TRUE`. If the `FastStartFailoverPmyShutdown` configuration property is set to `TRUE`, then the former primary database will have been automatically shut down and must be manually restarted before the master observer can attempt to reinstate it.

Note that these properties only affect whether primary shutdown and automatic reinstatement are performed if a fast-start failover occurs because the primary crashed or was isolated from the observer and target standby database.

See Also:

[Reinstating the Former Primary Database in the Broker Configuration](#) for more information about reinstatement

Restrictions When Fast-Start Failover is Enabled

This list describes restrictions when fast-start failover is enabled.

- Change:
 - The configuration protection mode
 - The redo transport mode used to send redo to the target standby database or the database currently in the primary role
 - The `FastStartFailoverTarget` configuration property on the primary unless the new property value contains the current fast-start failover target.
 - The `RedoRoutes` property on the primary if the new value would result in the primary not being able to ship redo to the current fast-start failover target standby. (It is permissible to change the `RedoRoutes` property on all standby databases including target standby databases.)
 - The `RedoRoutes` property on a far sync instance if it is being used to receive redo from the primary database and ship redo to the target standby database
- Disable or delete:
 - The broker configuration
 - The standby database that is the target of fast-start failover
 - A far sync instance if it is being used to receive redo from the primary database and ship redo to the target standby database
- Perform a manual failover:
 - Unless the conditions listed in [Performing Manual Role Changes When Fast-Start Failover Is Enabled](#) have been met
 - To a standby database that is not configured as the fast-start failover target

To determine if the configuration is ready for fast-start failover to occur, issue the `DGMGRL SHOW DATABASE <target-standby-database>` command, or query the `V$DATABASE` view on either the primary or target standby databases. The column value for `V$DATABASE.FS_FAILOVER_STATUS` will be `SYNCHRONIZED`

in a configuration operating in maximum availability mode, and it will be `TARGET UNDER LAG LIMIT` in a configuration operating in maximum performance mode when ready to fast-start failover. The `FS_FAILOVER_OBSERVER_PRESENT` column displays `YES` for the target standby database.

- Perform a switchover to a standby database that is not configured as the fast-start failover target
- Perform a switchover to the target standby database in a configuration operating in maximum availability mode, unless the standby database is synchronized with the primary database
- Perform a switchover to the target standby database in a configuration operating in maximum performance mode, unless the standby database is within the lag limit of the primary database

- Attempt to open the primary database, or the following error may be returned:

```
ORA-16649: possible failover to another database prevents this database from being opened
```

This error may return if the fast-start failover validity check fails or does not complete in under two minutes.

- Use the SQL `ALTER DATABASE MOVE DATAFILE` command to rename or relocate an online data file on a physical standby that is a fast-start failover target if the standby is mounted, but not open.

Configuring Fast-Start Failover in Observe-only Mode

The observe-only mode for fast-start failover enables you to test how fast-start failover will work in your environment. There is no impact on your current configuration or on applications.

In this mode, no actual changes are made to your Broker configuration. When the conditions for fast-start failover are met, the Broker adds messages to the observer log and broker log indicating that fast-start failover would have been initiated. The logs also contain other details about the actions that will be performed in case of a failover.

Note the following points about the observe-only mode:

- The primary database can enter `UNSYNC` or `LAGGING` state without an acknowledgement from the observer or target standby.
- The primary database can be opened even if there is no acknowledgement from the observer or target standby.
- Manual failover to the fast-start failover target can be performed without receiving an acknowledgement from the observer.
- Manual failover can be performed even if the pre-condition checks are not met. This includes the following: broker configuration is in `UNSYNC` or `LAGGING` state or unobserved state, failover target is invalid, reinstatement is in progress, or a master observer switch is in progress.
- You can switch over or manual failover to a bystander database.

To configure fast-start failover in observe-only mode:

```
DGMGRL> ENABLE FAST_START FAILOVER OBSERVE ONLY;
```

Shutting Down the Primary Database When Fast-Start Failover Is Enabled

Fast-start failover will not be triggered if the primary or standby database is shut down normally.

A normal shutdown uses `SHUTDOWN NORMAL`, `SHUTDOWN IMMEDIATE`, or `SHUTDOWN TRANSACTIONAL`. A normal shutdown prevents a fast-start failover until the primary database and standby database are connected and communicating again.

Performing Manual Role Changes When Fast-Start Failover Is Enabled

If fast-start failover is enabled you can still perform a switchover or a manual failover as long as certain conditions are met.

The conditions are as follows:

- The role change is directed to the same standby database that was specified for the `FastStartFailoverTarget` database property on the primary database.
- The target standby database is synchronized with the primary database if it is a configuration operating in maximum availability or maximum protection mode, or the target standby database is within the lag limit if it is a configuration operating in maximum performance mode.
- For manual failover, the observer is started and communicating with the target standby database. You must ensure that the primary database is shut down prior to performing a manual failover.

Note:

You can disable fast-start failover if necessary, by using the `FORCE` option. See [Disabling Fast-Start Failover](#).

See Also:

[Switchover](#) and [Manual Failover](#) for more information about switchovers and manual failovers, respectively

Directing a Fast-Start Failover From an Application

You can customize fast-start failover setup for a specific application by using the `DBMS_DG` PL/SQL package.

When a serious condition uniquely known to an application is detected, the application can call the `DBMS_DG.INITIATE_FS_FAILOVER` function to initiate an immediate fast-start failover. This function can be called from a connection to either the primary or any standby in the configuration. The database on which the procedure is called notifies the observer. The observer immediately initiates a fast-start failover, as long as the failover target database is in a valid fast-start failover state ("observed" and either "synchronized" or "within lag") to accept a failover. Once the observer has initiated a

fast-start failover, the primary database shuts down automatically. The observer does not attempt to reinstate the former primary database.

If the configuration is not failable, the `DBMS_DG.INITIALIZE_FS_FAILOVER` function returns an ORA error number (it does not signal an exception) informing the caller that a fast-start failover could not be performed.

 **Note:**

An application should use caution when calling the `DBMS_DG.INITIALIZE_FS_FAILOVER` function because the observer will initiate failover, if at all possible.

 **See Also:**

Oracle Database PL/SQL Packages and Types Reference for more information about the `DBMS_DG` package

Viewing Fast-Start Failover Configuration Statistics and Status

To verify the observer is started and the configuration is ready for fast-start failover, you can issue the `DGMGRL SHOW FAST_START FAILOVER`, `SHOW CONFIGURATION VERBOSE`, or `SHOW OBSERVER` commands.

Alternatively, you can query the `V$DATABASE` view on the target standby database.

You can also query the `V$FS_FAILOVER_STATS` view to display statistics about fast-start failover occurring on the system.

The rest of this section provides examples of using `DGMGRL SHOW` commands to display fast-start failover information and includes sections describing the following views:

- [V\\$DATABASE View](#)
- [V\\$FS_FAILOVER_STATS View](#)

Example 1: SHOW FAST_START FAILOVER

The `DGMGRL SHOW FAST-START FAILOVER` command displays all the fast-start failover related information. For example:

```
DGMGRL> SHOW FAST_START FAILOVER;

Fast-Start Failover: Enabled in Zero Data Loss Mode

Protection Mode: MaxAvailability
Lag Limit: 0 seconds

Threshold:          180 seconds
Ping Interval: 3000 milliseconds
Ping Retry: 0
Active Target:      South_Sales
Potential Targets: East_Sales, West_Sales
                   East_Sales    valid
```

```

    West_Sales    valid
Observer:        observer.example.com
Shutdown Primary:  TRUE
Auto-reinstate:   TRUE
Observer Reconnect: (none)
Observer Override: FALSE

```

Configurable Failover Conditions

```

Health Conditions:
  Corrupted Controlfile      YES
  Corrupted Dictionary       YES
  Inaccessible Logfile       NO
  Stuck Archiver             NO
  Datafile Write Errors      YES

```

```

Oracle Error Conditions:
  (none)

```

Example 2: SHOW CONFIGURATION VERBOSE

The following example shows the fast-start failover information for the DRSolution configuration:

```

Configuration - DRSolution

Protection Mode: MaxAvailability
Members:
North_Sales - Primary database
South_Sales - (*) Physical standby database

(*) Fast-Start Failover target

Properties:
  FastStartFailoverThreshold      = '60'
  OperationTimeout                = '30'
  TraceLevel                      = 'USER'
  FastStartFailoverLagLimit       = '30'
  CommunicationTimeout            = '180'
  ObserverReconnect               = '0'
  FastStartFailoverAutoReinstate  = 'TRUE'
  FastStartFailoverPmyShutdown    = 'TRUE'
  BystandersFollowRoleChange      = 'ALL'
  ObserverOverride                = 'FALSE'
  ExternalDestination1            = ''
  ExternalDestination2            = ''
  PrimaryLostWriteAction          = 'CONTINUE'
  ConfigurationWideServiceName    = 'North_Sales_CFG'

```

Fast-Start Failover: Enabled in Zero Data Loss Mode

```

Lag Limit: 0 seconds
Threshold: 30 seconds
Active Target: South_Sales
Potential Target: "South_Sales"
    South_Sales    valid
Observer: observer.example.com
Shutdown Primary: TRUE
Auto-reinstate: TRUE
Observer Reconnect: (none)
Observer Override: FALSE

```

```
Configuration Status:  
SUCCESS
```

Example 3: SHOW OBSERVER

The following `SHOW OBSERVER` command displays information about multiple observers in the `DRSolution broker` configuration.

```
DGMGRL> SHOW OBSERVER;  
  
Configuration - DRSolution  
  
Primary:                North_Sales  
Active Target Standby: South_Sales  
  
Observer "ob2" - Master  
  
  Host Name:                observer2.example.com  
  Last Ping to Primary:    1 second ago  
  Last Ping to Target:    2 seconds ago  
  
Observer "ob1" - Backup  
  
  Host Name:                observer1.example.com  
  Last Ping to Primary:    1 second ago  
  Last Ping to Target:    3 seconds ago  
  
Observer "ob3" - Backup  
  
  Host Name:                observer3.example.com  
  Last Ping to Primary:    4 seconds ago  
  Last Ping to Target:    5 seconds ago
```

V\$DATABASE View

You can query the `V$DATABASE` view to verify that the observer is started and the configuration is ready for fast-start failover.

When querying the `V$DATABASE` view, pay special attention to the following:

- The `FS_FAILOVER_STATUS` column, which can contain the values described in [Table 6-2](#). Note that if the `V$DATABASE.FS_FAILOVER_STATUS` column has a value of `DISABLED`, then any values returned for the remaining columns related to fast-start failover (`V$DATABASE.FS_FAILOVER_*`) become irrelevant.
- The `FS_FAILOVER_OBSERVER_PRESENT` column, which indicates whether the observer is running and actively pinging the database.

Table 6-2 FS_FAILOVER_STATUS Column of the V\$DATABASE View

Column Value	Description	Fast-Start Failover ...
BYSTANDER	Fast-start failover is enabled, but this standby database is not the target of the fast-start failover. The database cannot provide fast-start failover status information.	Is enabled
DISABLED	Fast-start failover is disabled.	Is not possible
LOADING DICTIONARY	Displays only on a logical standby database that has not yet completed loading a copy of the primary database's data dictionary.	Is not possible
PRIMARY UNOBSERVED	Displays only on the target standby database when it is SYNCHRONIZED with or is TARGET UNDER LAG LIMIT of the primary database, has connectivity to the observer, but the primary database does not have a connection to the observer.	Is not possible
REINSTATE FAILED	Reinstatement of the failed primary database as a new standby database failed. See Sources of Diagnostic Information for details about the broker's drc* log files.	Has completed
REINSTATE REQUIRED	The failed primary database requires reinstatement as a new standby database to the new primary. The observer automatically starts the reinstatement process. REINSTATE REQUIRED is present only after fast-start failover has occurred and shows on both the new primary database and the database undergoing reinstatement. This is cleared on both when the reinstatement has been completed.	Has completed

Table 6-2 (Cont.) FS_FAILOVER_STATUS Column of the V\$DATABASE View

Column Value	Description	Fast-Start Failover ...
STALLED	<p>Displays on the primary database after loss of connectivity to the target standby database and the change to the UNSYNCHRONIZED state (maximum availability mode) or to the TARGET OVER LAG LIMIT state (maximum performance mode) cannot be confirmed by either the target standby database or the observer. Note that the value of the <code>FastStartFailoverPmyShutdown</code> configuration property must be <code>FALSE</code> for the primary to stall indefinitely under these conditions. With a value of <code>TRUE</code> for this property, the primary will shut down after being stalled for the number of seconds specified by the <code>FastStartFailoverThreshold</code> property. It shuts down or stalls because it is likely a failover has occurred.</p> <p>Note: this state also occurs on the primary during startup when fast-start failover is possible and neither the target standby database nor the observer are present to confirm it is okay to continue opening the database.</p>	Is possible
TARGET OVER LAG LIMIT	<p>Displays if the standby database's redo applied point lags the primary database's redo generation point by more than the number of seconds specified by the <code>FastStartFailoverLagLimit</code> configuration property and the configuration is operating in maximum performance mode.</p>	Is not possible
TARGET UNDER LAG LIMIT	<p>Displays if the standby database's redo applied point does not lag the primary database's redo generation point by more than the number of seconds specified by the <code>FastStartFailoverLagLimit</code> configuration property and the configuration is operating in maximum performance mode.</p>	Is possible
SUSPENDED	<p>Displays only on the target standby database when either the primary or target standby database was shut down in a controlled fashion (using the <code>NORMAL</code>, <code>IMMEDIATE</code>, or <code>TRANSACTIONAL</code>, options, but not the <code>ABORT</code> option). Fast-start failover is inhibited in this case. <code>SUSPENDED</code> is cleared when connectivity with the primary database is restored.</p>	Is not possible

Table 6-2 (Cont.) FS_FAILOVER_STATUS Column of the V\$DATABASE View

Column Value	Description	Fast-Start Failover ...
SYNCHRONIZED	Displays when the primary and target standby databases are synchronized and the configuration is operating in maximum availability mode.	Is possible if the target standby database displays SYNCHRONIZED and the FS_FAILOVER_OBSERVER_PRESENT column displays YES
UNSYNCHRONIZED	Displays when the target standby database does not have all of the primary database redo data and the configuration is operating in maximum availability mode.	Is not possible
FS_FAILOVER_MODE	Displays the current fast-start failover mode.	The mode can have one of the following values: <ul style="list-style-type: none"> • DISABLED: Fast-start failover is disabled. • OBSERVE-ONLY: Fast-start failover is enabled in observe-only mode. • ZERO DATA LOSS: Fast-start failover is enabled with zero data loss. In this mode, the FastStartFailoverLogLimit configuration property is set to zero. • POTENTIAL DATA LOSS: Fast-start failover is enabled with some data loss. In this mode, the FastStartFailoverLogLimit configuration property is set to a non-zero value. Fast-start failover can incur data-loss within the time specified by FastStartFailoverLogLimit.

V\$FS_FAILOVER_STATS View

You can query the V\$FS_FAILOVER_STATS view on the primary database to display statistics about fast-start failovers that have occurred on the system.

It can be useful to perform such queries because fast-start failovers are fully automated and can occur at any time. Some of the statistics that can be monitored are as follows:

- **LAST_FAILOVER_TIME** that shows the timestamp of last fast-start failover
- **LAST_FAILOVER_REASON** that shows the reason for the last fast-start failover

The following is an example of querying the `V$FS_FAILOVER_STATS` view:

```
SQL> SELECT LAST_FAILOVER_TIME, LAST_FAILOVER_REASON FROM V$FS_FAILOVER_STATS;

LAST_FAILOVER_TIME
-----
LAST_FAILOVER_REASON
-----
-----
02/13/2007 16:53:10
Primary Disconnected
```

Disabling Fast-Start Failover

Disabling fast-start failover prevents the observer from initiating a failover to the target standby database.

If fast-start failover is disabled, then manual failover may still be possible. See [Manual Failover](#) for information about manual failover.



Note:

Disabling fast-start failover does not stop the observer. To stop the observer, see [Stopping the Observer](#).

To disable fast-start failover, use the Fast-Start Failover wizard in Cloud Control or the `DGMGRL DISABLE FAST_START_FAILOVER [FORCE]` command. The `FORCE` option disables fast-start failover on the database to which you are connected even when errors occur. Whether or not you need the `FORCE` option depends mostly on if the primary and target standby database have network connectivity:

- If the primary and target standby database have network connectivity, and the database to which you are connected has network connectivity with the primary database, the `FORCE` option has no effect. Simply use `DISABLE FAST_START_FAILOVER`. This method will disable fast-start failover on all databases in the broker configuration.
If errors occur during the disable operation, the broker returns an error message and stops the disable operation.
- If the primary and target standby databases do not have network connectivity or if the database to which you are connected does not have network connectivity with the primary database, consider using `DISABLE FAST_START_FAILOVER` *with* the `FORCE` option.

The broker may not be able to disable fast-start failover on all databases in the broker configuration when you issue the `DISABLE FAST_START_FAILOVER FORCE` command. As a result, there is no guarantee that the observer will not perform a fast-start failover to the target standby database if the observer determines that conditions warrant a failover. The following list indicates the extent to which fast-start failover is disabled in the broker configuration when the `DISABLE FAST_START_FAILOVER FORCE` command is issued on the primary database, target standby database, and a standby database that is not the fast-start failover target.

If you issue this command on:

- The target standby database when it does not have connectivity with the primary database, fast-start failover is disabled only on the target standby database. In this

case, the observer cannot perform a fast-start failover even if conditions warrant a failover. Disabling fast-start failover with the `FORCE` option when connected to the target standby database guarantees that fast-start failover will not occur.

When the primary database and the target standby database regain network connectivity, the broker will disable fast-start failover for the entire broker configuration.

- The primary database, it attempts to disable fast-start failover on as many databases in the configuration with which it has a network connection. If the primary database does not have connectivity with the target standby database, fast-start failover remains enabled on the target standby database and the observer may still attempt a fast-start failover if conditions warrant a failover.

▲ Caution:

This action may result in two databases in the configuration simultaneously assuming the primary database role should fast-start failover occur. For this reason, you should first issue this command on the target standby database.

- Another standby database that does not have connectivity with the primary database, fast-start failover is disabled for this database. Because fast-start failover was not disabled on the target standby database, the observer may still attempt a fast-start failover to the target standby database should conditions warrant a failover.

When the primary database and the (non-target) standby database regain network connectivity, the broker will propagate its current fast-start failover setting (`ENABLED` or `DISABLED`) to the non-target standby.

▲ Caution:

When you are experiencing network disconnections and you issue the `DISABLE FAST_START_FAILOVER FORCE` command on the primary database or a standby database that does not have connectivity with the primary database, fast-start failover may not be disabled for all databases in the broker configuration. As a result the observer may still initiate fast-start failover to the target standby database, if conditions warrant a failover. This may result in two databases in the configuration simultaneously assuming the primary database role.

Conditions Requiring the FORCE Option

Disabling fast-start failover without the `FORCE` option can succeed only if the database on which the command is issued has a network connection with the primary database and if the primary database and target standby database have a network connection. This is the recommended method for disabling fast-start failover.

However, there may be situations in which you must disable fast-start failover when the primary database and the target standby database do not have a network connection, or the database on which you issued the disable fast-start failover

command does not have a network connection to the primary database. In cases where there is a lost network connection, be aware that the observer may attempt a fast-start failover to the target standby database if conditions warrant a failover. The `FORCE` option may be the preferred method for disabling fast-start failover when:

- A network outage isolates the primary database from the observer and the target standby database before conditions exist that warrant a failover.

In this case, the primary database stalls and prevents any further transactions from committing because a fast-start failover may have occurred while it was isolated. If you expect the network to be disconnected for a long time and you need to make the primary database available, first confirm that a fast-start failover has not occurred to the target standby database. Then, disable fast-start failover with the `FORCE` option on the primary database. If possible, confirm that fast-start failover has not occurred to the target standby database prior to disabling fast-start failover with the `FORCE` option on the primary database.

▲ Caution:

This action may result in two databases in the configuration simultaneously assuming the primary database role. This can be avoided by first disabling fast-start failover with the `FORCE` option on the target standby.

- You want to conduct a manual failover to any standby database in the configuration (for example, because a failure occurred on the primary database at a time when the primary and target standby database were not ready to failover).

In this case fast-start failover cannot occur because the databases are not ready to failover. You cannot perform a manual failover to the target standby database for the same reason. To proceed, you must first disable fast-start failover using the `FORCE` option, and then perform a manual failover.

▲ Caution:

This action will result in loss of data and the possibility of two databases in the configuration simultaneously assuming the primary database role. This can be avoided by first disabling fast-start failover with the `FORCE` option on the target standby.

- A fast-start failover to the target standby database fails.

If the failover fails for any reason, it could leave the target standby database inoperable, regardless of whether the target standby database is ready to failover. If there is another standby database that is available for failover, you can perform a manual failover to that standby database after you first disable fast-start failover using the `FORCE` option on that standby database.

- You want to prevent fast-start failover from occurring because the primary database will resume service soon.

In this case, disable fast-start failover using the `FORCE` option on the target standby database. Once the primary database regains connectivity with the target standby database, fast-start failover will be disabled for all the databases in the configuration.

Disabling Fast-Start Failover Using Cloud Control

Click **Disable** in the Fast-Start Failover wizard. Then, click **Continue** to proceed to the next page. See the Cloud Control online help for more information.

Disabling Fast-Start Failover Using DGMGRL

Issue the `DISABLE FAST_START FAILOVER` command or the `DISABLE FAST_START FAILOVER FORCE` command. See the "[DISABLE FAST_START FAILOVER](#)" command in [Oracle Data Guard Command-Line Interface Reference](#) for more information.

Performance Considerations for Fast-Start Failover

This list contains some recommendations to obtain better performance when using fast-start failover.

- The failover time is dependent upon whether the target standby database (physical or logical standby database) has applied all of the redo data it has received from the primary database.
- Enabling fast-start failover in a configuration operating in maximum performance mode provides better overall performance on the primary database because redo data is sent asynchronously to the target standby database. Note that this does not guarantee no data will be lost.
- Fast-start failover is faster when you take steps to optimize recovery so that the application of redo data to the standby database is kept up to date with the primary database's rate of redo application. To optimize the log apply rate:
 - Do not configure the `DelayMins` database property to delay applying archived redo log files to the standby database (see [Managing Log Apply Services](#) for more information).
 - See *Oracle Data Guard Concepts and Administration* for information about tuning the log apply rate for a physical standby database.
 - See the *Oracle Maximum Availability Architecture* technical briefs at: <http://www.oracle.com/goto/maa>
- When setting the `FastStartFailoverLagLimit` configuration property, consider these tradeoffs between performance and potential data-loss:
 - A low lag limit will minimize data loss but may impact the performance of the primary database.
 - A high lag limit may lead to more data loss but may lessen the performance impact of the primary database.

Managing the Observer

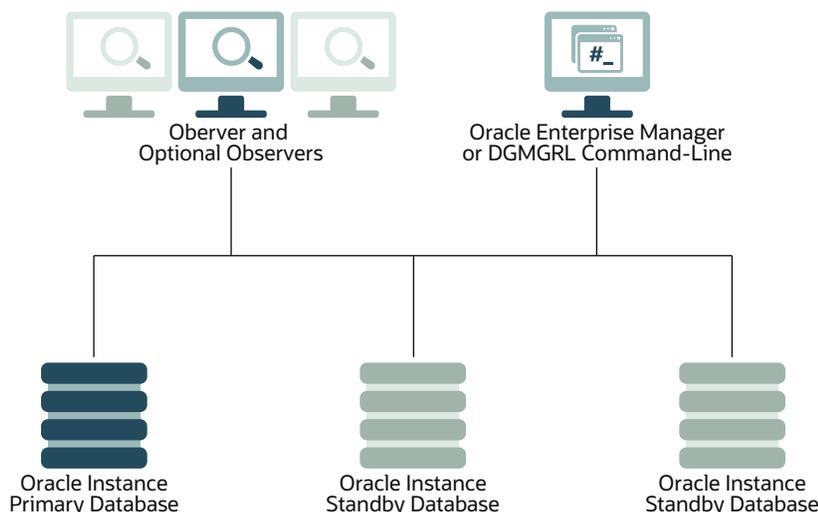
An observer process is integrated in the DGMGRL client-side component of the broker and typically runs on a different computer from the primary or standby databases and from the computer where you manage the broker configuration.

There can be up to four observers for a single Data Guard configuration. Observers continuously monitor the fast-start failover environment to ensure the primary database is available (described in [When Fast-Start Failover Is Enabled and the Observer Is Running](#)). The observer's main purpose is to enhance high availability and

lights out computing by reducing the human intervention required by the manual failover process that can add minutes or hours to downtime.

You can manage observers through either the Oracle Data Guard Overview pages in Cloud Control or using DGMGRL commands. [Figure 6-2](#) shows the observer monitoring a fast-start failover configuration.

Figure 6-2 The Observer in the Fast-Start Failover Environment



The following sections provide information about managing observers:

- [Installing and Starting the Observer](#)
- [Viewing Information About the Observer](#)
- [What Happens if the Observer Fails?](#)
- [Stopping the Observer](#)
- [Moving the Observer to Another Computer](#)
- [How the Observer Maintains Fast-Start Failover Configuration Information](#)
- [Patching an Environment When the Observer Is Running and Fast-start Failover Is Enabled](#)

Installing and Starting the Observer

Observers should be installed and run on a computer system that is separate from the primary and standby systems.

Installing and starting an observer is an integral part of using fast-start failover and is described in detail in the following sections:

- [Oracle Data Guard Installation](#) explains that you can either install only the Oracle Client Administrator or you can install the complete Oracle Database Enterprise Edition or Personal Edition on the observer system.
- [Enabling Fast-Start Failover](#) describes how to start observers as a part of the step-by-step process to enable fast-start failover. Examples of starting observers using DGMGRL are included in [Scenario 6: Enabling Fast-Start Failover and Starting the Observer](#).

To start an observer, you must be able to log in to DGMGRL with an account that has the `SYSDG` or `SYSDBA` privilege. An observer is an OCI (Oracle Call Interface) client that connects to the primary and target standby databases using the same `SYS` credentials you used when you connected to the Oracle Data Guard configuration with DGMGRL.

See Also:

- The My Oracle Support note 1625597.1 at <http://support.oracle.com> for information about compatibility requirements between the observer and DGMGRL

Starting Multiple Observers on a Data Guard Broker Configuration

Note:

For Oracle Database Release 12.2 and higher, Oracle Enterprise Manager Cloud Control (Cloud Control) supports configuring multiple observers using the Enterprise Manager Command Line Interface (EM CLI). Use the EMCLI verb `dg_configure_observers`. The subcommands for this verb include `start`, `stop`, `setMaster`, `show`, and `delete_alternate_observer`. See Oracle Enterprise Manager Command Line Interface.

You can register up to four observers to monitor a single Data Guard broker configuration. Each observer is identified by a name that you supply when you issue the `START OBSERVER` command. The following conditions apply when multiple observers are registered for one configuration:

- When fast-start failover is enabled, one of the observers is the master observer. The remaining observers are called backup observers. When fast-start failover is disabled, no observer is called the master observer; all observers have the same functionality.
- No two observers on the same Data Guard Broker configuration can have the same name.
- If no name is specified for the observer then a default observer name, the host name of machine where the `START OBSERVER` command is issued, is used.
- An observer name is case-insensitive.
- The string "NONAME" cannot be used as an observer name.
- Each observer has its own log file. The log file name is specified with the `LOGFILE IS` option of the `START OBSERVER` command. If the specified log file is not accessible, or the `LOGFILE IS` option is not used, then the observer output is sent to standard output.
- Except for testing purposes, it is not recommended that you start more than one observer on the same host for a Data Guard broker configuration.
- The command `SHOW FAST_START FAILOVER` shows a list of registered observers and indicates which one is the master.

- The command `SHOW OBSERVER` provides detailed information about registered observers.

The Master Observer and Backup Observers

Note:

The master observer works in the same manner that a single observer worked prior to the advent of multiple observers in Oracle Database 12c Release 2 (12.2.0.1).

The observer does not need to coordinate fast-start failover when fast-start failover is disabled, so the primary and target standby do not nominate a master observer until fast-start failover is enabled. When fast-start failover is enabled, the primary and standby randomly choose one of the registered observers to be the master. If there are no registered observers when fast-start failover is enabled, then the first observer started is designated as the master observer, and all others started later are backup observers. Only the master observer can coordinate fast-start failover with Data Guard broker. All other registered observers are considered to be backup observers.

Manually Changing the Master Observer

If necessary, you can use the `SET MASTEROBSERVER TO Observer-Name` command to change which observer is the master. For example:

```
DGMGRL> SET MASTEROBSERVER TO boston-observer;  
Succeeded.
```

If the `PreferredObserverHosts` property is set for the current primary, only the observers specified in the property can be used in the `SET MASTEROBSERVERHOST TO` command. When this command is issued, the actual switch does not happen until the next time the primary contacts the target standby, usually within three seconds if fast-start failover is enabled. You can issue a `SHOW OBSERVER` command to confirm that the switch took place. For more information, see [SET MASTEROBSERVER TO](#).

Starting Observers as Background Processes

To run an observer as a background process, use the DGMGRL command `START OBSERVER IN BACKGROUND`. You must provide a connect identifier, through which one or more databases in a specific broker configuration can be reached. The following is an example of starting an observer as a background process:

```
DGMGRL> START OBSERVER IN BACKGROUND  
FILE IS /net/sales/dat/oracle/broker/fsfo.dat  
LOGFILE IS /net/sales/dat/oracle/broker/observer.log  
CONNECT IDENTIFIER IS sales_p
```

```
Submitted command "START OBSERVER" using connect identifier "sales_p"
```

```
DGMGRL>
```

The `START OBSERVER IN BACKGROUND` command uses Oracle wallet to obtain credentials to log into the database server and register observers. Even if you have successfully connected to a database server in the broker configuration using the `CONNECT` command, this command

ignores the existing connection and uses the credentials stored in Oracle wallet. For more information, see [START OBSERVER IN BACKGROUND](#).

Starting Multiple Observers On a Single Host

If you want to use one Oracle home to start multiple observers, with each observer monitoring a different fast-start failover configuration, use the `FILE` qualifier to specify a unique observer configuration file location for each configuration to be monitored. If you want to capture any logging generated by the observer, use the `LOGFILE IS` option on the `START OBSERVER` command, and ensure that the file name is unique. For example:

```
% dgmgrl
DGMGRL> CONNECT /@primary1;
DGMGRL> START OBSERVER FILE IS '/u01/oracle/dbs/config1.dat'
LOGFILE IS '/u01/oracle/log/config1.log'

% dgmgrl
DGMGRL> CONNECT /@primary2;
DGMGRL> START OBSERVER FILE IS '/u01/oracle/dbs/config2.dat'
LOGFILE IS '/u01/oracle/log/config2.log'
```

Viewing Information About the Master Observer

You can find information about the master observer by querying the `V$DATABASE` view.

- `FS_FAILOVER_OBSERVER_HOST` shows the name of the computer on which the master observer is running
- `FS_FAILOVER_OBSERVER_PRESENT` shows whether or not the master observer is connected to the local database

The Column Value in the following table is consistent across instances in an Oracle Real Applications Clusters (Oracle RAC) environment. That is, if the observer is connected to any instance in the Oracle RAC, all instances will show a value of `YES`.

Table 6-3 FS_FAILOVER_OBSERVER_PRESENT Column of the V\$DATABASE View

Column Value	Description
YES	The master observer is currently connected to the local database
NO	The master observer is not connected to the local database

For example, to determine if fast-start failover can occur, the `FS_FAILOVER_STATUS` column displays either `SYNCHRONIZED` or `TARGET UNDER LAG LIMIT` and the `FS_FAILOVER_OBSERVER_PRESENT` column displays `YES` for the target standby database. For example:

Database	FS_FAILOVER_STATUS	Protection Mode	FS_FAILOVER_OBSERVER_P RESENT
Primary	SYNCHRONIZED	Maximum Availability	YES

Database	FS_FAILOVER_STATUS	Protection Mode	FS_FAILOVER_OBSERVER_P RESENT
Standby	SYNCHRONIZED	Maximum Availability	YES
Primary	TARGET UNDER LAG LIMIT	Maximum Performance	YES
Standby	TARGET UNDER LAG LIMIT	Maximum Performance	YES

In the following example, assume the network between the primary database and the observer has failed. In this case, the `FS_FAILOVER_STATUS` and `FS_FAILOVER_OBSERVER_PRESENT` columns will appear as shown in the following table and fast-start failover will not occur:

Database	FS_FAILOVER_STATUS	FS_FAILOVER_OBSERVER_PRESENT
Primary	SYNCHRONIZED	NO
Standby	PRIMARY UNOBSERVED	YES

See Also:

- *Oracle Database Reference* for more information about the `V$DATABASE` view.

Viewing Information About All Observers

You can find detailed information about all observers, including master observers and backup observers, in the `V$FS_FAILOVER_OBSERVERS` view.

For each observer, the `V$FS_FAILOVER_OBSERVERS` view provides the observer name, host, whether it is the master observer, when it became the master observer, whether it is currently connected to the primary and target standby databases, the location of the observer log file, and the location of the observer runtime data file (`fsfo.dat`).

See Also:

- *Oracle Database Reference* for more information about the `V$FS_FAILOVER_OBSERVERS` view.

What Happens if the Master Observer Fails?

The behavior of the broker if the master observer fails depends on whether the broker configuration has one observer or multiple observers.

If there is only one observer, then it is considered to be the master observer. If there are multiple observers, then only one of them is the master observer.

Broker Configuration Has Only One Registered Observer

When the configuration has only one registered observer, if the primary and target standby databases stay connected but the connection to the observer is lost, then the broker reports that the configuration is not observed. The configuration and database status report that the observer is not running and return one of the following status messages:

```
ORA-16658: unobserved fast-start failover configuration
ORA-16820: fast-start failover observer is no longer observing this database
```

While the configuration is in the unobserved state, fast-start failover cannot happen. Therefore, the primary database can continue processing transactions, even if the target standby database fails. The configuration status returns the `SUCCESS` status after the observer reestablishes its connection to the primary database, which then notifies the target standby database.



See Also:

[Viewing Fast-Start Failover Configuration Statistics and Status](#)

Broker Configuration Has Multiple Registered Observers

When the configuration has more than one registered observer, if the primary and target standby databases stay connected but the connection to the master observer is lost, then the broker tries to nominate a backup observer as the new master observer. This nomination is noted in the observer log file and in the broker log file (`drc*.log`). Additionally, the new master observer is identified in the output shown for the `SHOW FAST_START_FAILOVER` and `SHOW OBSERVER` commands.



Note:

During the period when a new master observer is being identified, a fast-start failover cannot occur until the new master observer is selected and observing the configuration.

If the primary or target standby databases lose connections to all backup observers, then the broker does not try to nominate a backup observer as the new master observer, and the broker reports that the configuration is not observed. The configuration and database status report the same error messages as are returned when there is only one registered observer.

Managing Observer's Connection to the Primary

The `ObserverOverride` and `ObserverReconnect` properties allow you additional control over the connection to the primary.

When the observer loses its connection to the primary database for a period of time greater than that specified by the `FastStartFailoverThreshold` property, it attempts a failover to the standby database. However, if the standby has had contact from the

primary within the period of time specified by the `FastStartFailoverThreshold` property, the standby prevents the failover attempt.

To override this behavior and allow a fast-start failover to occur if the observer is unable to contact the primary for more than `FastStartFailoverThreshold` seconds, set the `ObserverOverride` property to `TRUE`. For example:

```
DGMGRL> EDIT CONFIGURATION SET PROPERTY ObserverOverride=TRUE;
```

Ordinarily the observer connects once to the primary and does not attempt to reconnect unless the connection has failed. However, if you want the observer to reconnect to the primary database periodically as a means of testing the health of the network connection to the primary, then use the `ObserverReconnect` configuration property. This specifies how often the observer establishes a new connection to the primary database. In the following example, `ObserverReconnect` is set to 30 seconds. This results in the observer establishing a new connection to the primary database every 30 seconds.

```
DGMGRL> EDIT CONFIGURATION SET PROPERTY ObserverReconnect=30;
```

Configuring the Time Taken to Initiate Fast-Start Failover

Use broker configuration properties to set the time taken to detect a failure on the primary database.

Before enabling fast-start failover, use one of the following techniques to set the time taken to detect a failure on the primary database:

- Set the `FastStartFailoverThreshold` configuration property
The broker initiates a failover after the number of seconds specified by this property. The minimum detection time is 6 seconds, which is the default value of the `FastStartFailoverThreshold` property.
- Set the `ObserverPingInterval` and `ObserverPingRetry` configuration properties
Set both these properties to achieve a primary failure detection time of 1 second. The minimum value of `ObserverPingInterval` is 100 milliseconds and that of `ObserverRetryCount` is 10.



Note:

Setting the `ObserverPingInterval` and `ObserverPingRetry` properties must be used only in low latency environments.

Stopping the Observer

You can manually stop a specific observer or all observers.

Before stopping an observer, note the following:

- The observer does not stop immediately when you issue the `STOP OBSERVER` command. After the broker receives the `STOP OBSERVER` request, the request is passed to the observer the next time the observer contacts the broker, and the observer then stops itself.

- Stopping the observer does not disable fast-start failover. However, fast-start failover cannot occur when the target standby database is in an unobserved state.
- To stop the observer when fast-start failover is disabled, the primary database must be running.
- To stop the observer when fast-start failover is enabled, the primary database and target standby database must be connected and communicating with each other. If they are isolated from each other, then you must first disable fast-start failover by using the `FORCE` option, and then stop the observer. (See [Disabling Fast-Start Failover](#) for important considerations when using the `FORCE` option.)

Stopping All Observers

You can specify `STOP OBSERVER ALL` to stop all observers registered in a broker configuration.

Stopping a Specific Observer When There are Multiple Observers

To stop a specific observer when there are multiple registered observers running, issue the following command:

```
DGMGRL> STOP OBSERVER observer_name;
```

You can log into DGMGRL from any machine to stop an observer. For instance, you could log into the system running `observer1` to stop `observer2`.

In an environment where there are multiple observers configured, stopping the master observer is not allowed unless it is the last running observer. To stop an observer currently designated as the master observer, first issue the `SET MASTEROBSERVER` command to designate a different observer as master observer. Then the `STOP OBSERVER` command can be issued successfully on the former master observer.

Stopping the Observer When There is Only One Observer

If there is only one registered observer, then it works in the same manner that a single observer worked prior to the advent of multiple observers in Oracle Database 12c Release 2 (12.2.0.1). To stop it, you can do either of the following:

- Using Cloud Control
Choose the **Stop Observer** option on the first page of the fast-start failover wizard and click **Continue** at the bottom of the page. See the Cloud Control online help for more information.

- Using DGMGRL

Issue the following command:

```
DGMGRL> STOP OBSERVER;
```

If there is more than one registered observer, then this command returns an error; a name is required if there is more than one observer.

**Note:**

See the [STOP OBSERVER](#) command for more information.

Moving the Observer to Another Computer

An observer can be moved from one computer to another through a process of stopping it on one system and re-starting it on another.

To move the observer to another computer:

1. Stop the observer from any computer system in the broker configuration, as described in [Stopping the Observer](#).
2. Start the observer on the new computer system, as described in Step 8 of [Enabling Fast-Start Failover](#).

There is no need to disable fast-start failover when you move the observer.

How the Observer Maintains Fast-Start Failover Configuration Information

The observer persistently maintains information about the fast-start failover configuration in a binary file created in the working directory where you started the observer.

By default, the observer creates this file in the current working directory when it is started and names the file `fsfo.dat`. This file contains connect identifiers to both the primary and the target standby databases.

Ensure this file cannot be read by unauthorized users.

Once the observer is started, you cannot change the file's name and location. However, you can change the name or the location of the file if you start the observer using the `DGMGRL START OBSERVER` command and include the `FILE IS` qualifier. See the [START OBSERVER](#) command for more information.

**Note:**

If the observer is stopped abnormally (for example, by typing `CTRL/C`), restart it and reference the existing `fsfo.dat` file with the `FILE IS` qualifier.

Managing Observers for Multiple Configurations

DGMGRL can be used to manage multiple observers in a group of broker configurations.

You can start, stop, and show observers for a group of configurations. You can also switch the master observer hosts for a group of configurations to one specific host.

The group of broker configurations to be managed is declared in the observer configuration file. The default group is all the configurations defined in the observer configuration file. If you do not want to use the default, you can define a specific group. The observer configuration file is a text file and the syntax to define observers and groups is similar to that used in the `listener.ora` or `tnsnames.ora` files. It has two parts in the following order:

1. Configuration declaration — this section is mandatory. It must appear as the first part of an observer configuration file.
2. Group definition — this section is optional. If provided, then each group needs to have at least one broker configuration declared in the second part.

Syntax for Mandatory Configuration Declaration

The syntax for the configuration declaration is:

```
BROKER_CONFIGS = (  
  (configuration1-definition)  
  (configuration2-definition)  
  ...  
  (configuration-n-definition)  
)
```

The definition for each broker configuration is:

```
CONFIG= (NAME           =configuration-name)  
        (CONNECT_ID    =connect-identifier)  
        (CONFIG_HOME   =config-home-location)  
)
```

Note the following:

- The `configuration-name` can be different from the name defined in the metadata of the Data Guard Broker configuration. (This is useful because the name defined in the metadata may contain whitespace and international characters, which the observer configuration file does not allow.)
- The `connect-identifier` is a TNS alias defined in `tnsnames.ora` through which all instances of all databases in this Data Guard broker configuration can be reached.
- `Config-home-location` specifies a location for the observer log file and `FSFO.dat` file.
- Duplicate configuration names in configuration definitions are not allowed.

Syntax for Optional Group Definition

The syntax for the optional definition of a broker configuration group is:

```
CONFIG_GROUPS = (  
  (group1-definition)  
  (group2-definition)  
  ...  
  (group-n-definition)  
)
```

The definition of each group is:

```
GROUP = (NAME=group-name)  
        (CONFIG_LIST=  
          (NAME=configuration1-name)
```

```

        (NAME=configuration2-name)
        ...
        (NAME=configuration-n-name)
    )

```

Note the following:

- The group definition section is optional. If groups are not defined, you can still operate on all configurations defined in the file as a whole.
- The word `ALL` cannot be used as a group name because it is a reserved keyword.
- Duplicate group names are not allowed.
- Each group that you define must have at least one broker configuration.
- Any broker configuration name that is referred to must exist in the configuration declaration section.
- A broker configuration can belong to multiple groups.

Specifying the Observer Configuration File

When you execute commands that affect multiple observers, if you have not specified a name and location for the observer configuration file, then broker searches the current working directory for a file named `observer.ora`. The command fails if the file does not exist.

However, you do have the option of specifying a name and location for the observer configuration file. To do this, use the `SET ObserverConfigFile` and `SHOW ObserverConfigFile` commands. These commands can be issued from the `DGMGRL` command line, but it is not necessary to log on prior to using them.

- `SET ObserverConfigFile` — used to specify a customized observer configuration file. `ObserverConfigFile` is a `DGMGRL` session runtime property. You must specify it every time you start a new `DGMGRL` client.
- `SHOW ObserverConfigFile` — used to check the runtime property `ObserverConfigFile`. If you have not used the `SET ObserverConfigFile` command after starting the current `DGMGRL` client, then the result will always be: `ObserverConfigFile=observer.ora`.

Commands For Managing Observers on Multiple Configurations

The commands that can be executed for a group of configurations (as declared in an observer configuration file) are as follows.

- `START OBSERVING [cfg_group_name]` — starts a new observer for each broker configuration in the specified group. If the group name is not provided, then a new observer is started for each broker configuration defined in `observer.ora`.
- `STOP OBSERVING [cfg_group_name]` — stops `LOCAL` observers running on this host (where this `DGMGRL` is running) for all broker configurations in a specified group
- `SHOW OBSERVERS [FOR fg_group_name]` — shows information about observers for all configurations in the specified group. The information shown by this command is the same as that shown by a `SHOW OBSERVER` command on each individual configuration. If a group name is not specified, then `SHOW OBSERVERS` alone is also a valid command. It behaves similarly to `START OBSERVING` and `STOP OBSERVING` to operate on all the configurations defined in the observer configuration file.

- SET MASTEROBSERVER TO — allows you to manually change the observer configuration file.



See Also:

- [Oracle Data Guard Command-Line Interface Reference](#) for more information about these broker commands.

Credentials Required for Access to Broker Configurations

You must use the Oracle wallet to store the credentials for all broker configurations to be managed. Data Guard broker does not manage or store credentials. It uses the connect identifier specified in the observer configuration file to locate the credentials for a broker configuration from the Oracle wallet. If the credentials cannot be obtained, then the attempted command fails (but only for the broker configuration whose credentials have not been obtained).

Example 6-2 Sample Observer Configuration File

The following is a sample observer configuration file:

```
# is the comment sign in this file
BROKER_CONFIGS = (
  #Broker configuration 'SALES'; the connect ID is 'SALES_P'
  (CONFIG = (NAME=SALES)
    (CONNECT_ID=SALES_P)
    (CONFIG_HOME= ORACLE_HOME/LOG/))
  #Broker configuration 'HR'; the connect ID is 'HR_P'
  (CONFIG = (NAME=HR)
    (CONNECT_ID=HR_P)
    (CONFIG_HOME= ORACLE_HOME/LOG/))
  #Broker configuration 'CUSTOMER'; The connect ID is 'CUSTOMER_P'
  (CONFIG = (NAME =CUSTOMER)
    (CONNECT_ID=CUSTOMER_P)
    (CONFIG_HOME= ORACLE_HOME/LOG1/))
)
# The portion 'CONFIG_GROUP' lists definitions of
# all broker configuration groups
CONFIG_GROUPS= (
  # Configuration 'GRP_A' includes 'SALES' and 'HR'
  (GROUP = (NAME=GRP_A)
    (CONFIG_LIST=
      (NAME=SALES)
      (NAME=HR)))
  # Configuration 'GRP_B' includes 'SALES' and 'CUSTOMER'
  (GROUP = (NAME=GRP_B)
    (CONFIG_LIST=
      (NAME=SALES)
      (NAME=CUSTOMER)))
)
```

Since the broker configuration SALES consists of three databases, Boston, Chicago, and Dallas, with a CONNECT_ID of SALES_P, the SALES_P connect identifier must be defined such that it can reach any instance of any database within the configuration. The new ConfigurationWideServiceName configuration property can be used to simplify setting up this connect identifier. Data Guard broker publishes this service on each instance as it comes up and broker management of the instance is initialized:

```
SALES_P = (DESCRIPTION =
            (ADDRESS_LIST =
              (ADDRESS=(PROTOCOL=TCP) (HOST=boston-cluster) (PORT =
1521))
              (ADDRESS=(PROTOCOL=TCP) (HOST=chicago-cluster) (PORT =
1521))
              (ADDRESS=(PROTOCOL=TCP) (HOST=dallas-cluster) (PORT =
1521)))
            (CONNECT_DATA=(SERVER=DEDICATED)
(SERVICE_NAME=cfgSvc.us.example.com)))
```

Patching an Environment When the Observer Is Running and Fast-start Failover Is Enabled

To patch an environment where the Observer is running and fast-start failover is enabled, follow these steps prior to applying the patch.

1. Stop the observer using the DGMGRL STOP OBSERVER command. If multiple observers have been started for the configuration, then be sure to specify the name of the observer whose environment is to be patched (STOP OBSERVER *observer-name*). The primary and target standby must have connectivity for the STOP OBSERVER command to complete successfully.
2. Disable fast-start failover using the DGMGRL DISABLE FAST_START FAILOVER command. Note the primary and target standby must have connectivity for this command to complete successfully.

Note:

It is not necessary to disable fast-start failover if there are only backup observers, but no master observer, running in the environment to be patched.

After the patch has been successfully applied to all databases, take the following steps to enable fast-start failover and start the observer.

1. Enable fast-start failover using the DGMGRL ENABLE FAST_START FAILOVER command. The primary and target standby must have connectivity for this command to complete successfully.
2. Start the observer using the DGMGRL START OBSERVER command. If the configuration has multiple observers, then be sure to specify the name of the observer whose environment has just been patched (START OBSERVER *observer-name*).

Reinstating the Former Primary Database in the Broker Configuration

Reinstatement restores high availability to the broker configuration so that, in the event of a failure of the new primary database, another fast-start failover can occur.

If a fast-start failover was initiated because the primary database had crashed or lost connectivity with the master observer and target standby database, then the master observer automatically attempts to *reinst*ate the former primary database as a standby database, if the `FastStartFailoverAutoReinst` configuration property is set to `TRUE`. The reinstated database acts as the fast-start failover target for the new primary database, making a subsequent fast-start failover possible. The new standby database is a viable target of a failover when it begins receiving redo data received from the new primary database.

To allow the master observer to automatically reinstate the former primary database, the database must be started and mounted. It will not be allowed to open in any event if fast-start failover is enabled. The broker reinstates the database as a standby database of the same type as the former standby database of the new primary database.

If the former primary database cannot be reinstated automatically, you can manually reinstate it using either the `DGMGRL REINSTATE` command or Cloud Control. Step-by-step instructions for manual reinstatement are described in [Reenabling Disabled Databases After a Role Change](#).

Requirements

Reinstatement is supported only after failover in a broker configuration. It also requires Flashback Database to be enabled on both the primary and target standby databases.

[Prerequisites for Enabling Fast-Start Failover](#) provides complete information about all of the fast-start failover and reinstatement requirements.

Restrictions on Reinstatement

This list describes conditions in which the broker cannot automatically reinstate the former primary database.

- A fast-start failover occurred because a user-configurable condition was detected or was requested by an application by calling the `DBMS_DG.INITIATE_FS_FAILOVER` function.
- `FastStartFailoverAutoReinst` is set to `FALSE`
- Another failover or switchover occurred *after* the fast-start failover completed but *before* the former primary database restarted
- Fast-start failover was disabled
- The master observer cannot connect to the former primary database
- The former primary database cannot connect to the new primary database
- The former primary database and the new primary database are not configured in the same fast-start failover environment
- The former primary database was disabled because of a manual failover when fast-start failover was disabled

 **Note:**

Standby databases that are disabled during switchover, manual failover, or fast-start failover will not be automatically reinstated.

If automatic reinstatement fails, the broker will log errors and the former primary database will remain in the mounted state. At this point, you can either:

- Disable fast-start failover (described in [Disabling Fast-Start Failover](#)) and attempt to open the former primary database
- Manually reinstate the former primary database, as described in [Reenabling Disabled Databases After a Role Change](#)

How the Broker Handles a Failed Reinstatement

If a failure occurs once a reinstatement operation (automatic or manual) is underway, the broker logs the appropriate information in the broker configuration files and broker log files.

The former primary database is disabled. Most in-progress failures cannot be restarted (for example, archived redo log file corruption on the primary database). You must manually re-create the database as a standby database and then reenoble it.

 **See Also:**

[How to Re-create and Reenable a Disabled Database](#)

Shutting Down Databases In a Fast-Start Failover Environment

If necessary, you can shut down the primary or target standby database in a fast-start failover environment.

Perform the following steps:

1. Stop the observer and wait for the `REGISTERED` column of all rows in the `V$FS_FAILOVER_OBSERVERS` fixed view to contain the value `NO` for both the primary and target standby databases. This ensures that a fast-start failover will not occur while you are shutting down the primary database.
2. Shut down the primary database and the target standby database using either `DGMGRL SHUTDOWN` command or the SQL*Plus `SHUTDOWN` statement.

When restarting the databases, you may restart them in any order. When both databases have been restarted, you may restart the observer.

Bystander standby databases can be shut down at any time in any order without impacting fast-start failover.

Database Client Considerations

Event notification and database connection failover support is available to database clients connected to local database services when a broker-managed failover occurs.

For information about event notification and database connection failover support for global services, see the *Oracle Database Global Data Services Concepts and Administration Guide*.

After a failover, the broker publishes Fast Application Notification (FAN) events. These FAN events can be used in the following ways:

- Applications can use FAN without programmatic changes if they use one of these Oracle integrated database clients: Oracle Database JDBC, Oracle Database Oracle Call Interface (OCI), Oracle Data Provider for .NET (ODP.NET), or Universal Connection Pool for Java. These clients can be configured for Fast Connection Failover (FCF) to automatically connect to a new primary database after a failover.
- JAVA applications can use FAN programmatically by using the JDBC FAN application programming interface to subscribe to FAN events and to execute event handling actions upon the receipt of an event.
- FAN server-side callouts can be configured on the database tier.

FAN events are published using Oracle Notification Services (ONS) for all Oracle integrated database clients in Oracle Database 12c and later. In previous releases, OCI and ODP.NET clients receive FAN notifications via Oracle Advanced Queuing (AQ).

 **Note:**

A single-instance database must be registered with Oracle Restart in order to publish FAN events via ONS.

 **See Also:**

- *Oracle Real Application Clusters Administration and Deployment Guide* for more information about configuring FAN, FCF, and ONS on an Oracle Real Application Clusters (Oracle RAC) database

Oracle Data Guard Specific FAN and FCF Configuration Requirements

There are configuration requirements that must be met in order to publish and properly handle FAN events generated as the result of a broker-managed failover.

These requirements are supplemental to those described in the documents previously referenced and in the following client-specific guides:

- *Oracle Database JDBC Developer's Guide*
- *Oracle Call Interface Programmer's Guide*
- *Oracle Data Provider for .NET Developer's Guide for Microsoft Windows*

Oracle Net Configuration Requirements

For Fast Connection Failover (FCF) to occur, a client must be able to locate the new primary database after a failover.

This section describes how to configure an Oracle Net connect descriptor that meets this requirement.

The connect descriptor can be configured in one of two ways:

1. Configure the connect descriptor for connect-time failover. Add the primary database and each standby database to the address list. Set the `CONNECT_TIMEOUT` parameter to a small value to minimize the delay experienced if a network address is not available. Increase the value of this parameter if resource contention causes connection timeouts to occur during normal operation.

For example:

```
sales =
  (DESCRIPTION=
    (FAILOVER=ON)
    (CONNECT_TIMEOUT=5)
    (ADDRESS_LIST=
      (ADDRESS=(HOST=boston-scan) (PORT=1521))
      (ADDRESS=(HOST=dallas-scan) (PORT=1521)))
    (CONNECT_DATA=(SERVICE_NAME=sales)))
```

2. Configure the connect descriptor with a single network name that is registered with a global naming service such as DNS or LDAP. Create a trigger based on the `DB_ROLE_CHANGE` system event that changes the network address associated with the network name to the network address of the new primary database after a failover.

See Also:

- *Oracle Database PL/SQL Language Reference* for more information about the `DB_ROLE_CHANGE` system event

The connect descriptor must contain the `SERVICE_NAME` parameter in either case.

Database Service Configuration Requirements

Database services can be configured to be active in specific database roles on Oracle RAC databases and on single-instance databases managed by Oracle Restart.

The broker interacts with Oracle Clusterware or Oracle Restart to ensure that the appropriate database services are active and that the appropriate FAN events are published after a role change.

FAN events are always published through ONS. However, the event notifying a failover is only published for database services that have been configured to be active while the database is in the primary role on the new primary database.

Services that must be active in any given database role (primary, physical standby, logical standby, or snapshot standby) must be configured with the Server Control utility (SRVCTL) explicitly on each database where the service must be active.

In the following example commands, a service named `PAYROLL` is configured to be active in the `PRIMARY` role on the primary database `NORTH`. The service is then configured to be active in the `PRIMARY` role on the standby database `SOUTH`, so that it will be active on that database after a role transition. In these sample commands, the ellipse (...) signifies any other `add service` options you wish to supply.

 **Note:**

The examples shown in this section do not necessarily show the specific attributes you might need to use in your own environment. The required attributes vary depending on your configuration (including whether your environment is Oracle RAC-based or single-instance). Refer to the appropriate Oracle RAC or Oracle Restart documentation for further information.

On primary database `NORTH`, execute the following:

```
srvctl add service -db NORTH -service PAYROLL -role PRIMARY ...
```

On standby database `SOUTH`, execute the following:

```
srvctl add service -db SOUTH -service PAYROLL -role PRIMARY ...
```

Services that are to be active while the database is in the physical standby role must also be created and started on the current primary database regardless of whether the service will be started on that database or not. This is to ensure that the service definition gets propagated to the physical standby database via the redo stream and thus allows for the service to be started on the physical standby database. The service can be started on the physical standby only after the redo generated by starting the service has been applied. It is important that all `SRVCTL add service` options be identical on all the databases so that the services behave the same way before and after a role change.

If all the databases do not have the same values, `SRVCTL` attempts to override the values, which will fail on the physical standby database because it is open read-only. In the following example, a service named `sales` is configured to be active in the `PHYSICAL_STANDBY` role on the primary database `NORTH`. It is then started and stopped on the primary database. It could optionally also be removed from the primary database if there is no intention to ever run this service on the current primary database. It is then configured to be active in the `PHYSICAL_STANDBY` role on the physical standby database `SOUTH`.

Execute the following on primary database `NORTH`:

```
srvctl add service -db NORTH -service SALES -role PHYSICAL_STANDBY ...
```

```
srvctl start service -db NORTH -service SALES
```

```
srvctl stop service -db NORTH -service SALES
```

Execute the following on the physical standby database `SOUTH`:

```
srvctl add service -db SOUTH -service SALES -role PHYSICAL_STANDBY ...
```

If the broker now performs a switchover or failover, it automatically starts the `SALES` service on the correct database, based on the database's role.

The previous examples dealt with setting up only one service on a database. The following example shows you how to set up more than one service on a database and how using the broker ensures that the correct service starts on the correct database.

Suppose you have a primary database, `BOSTON`, and a standby database, `CHICAGO`. Issue the following `SRVCTL` commands so that both databases in the Data Guard configuration know about the two potential services for each database:

On `BOSTON`

```
srvctl add service -db BOSTON -service SALESRW -role PRIMARY -y AUTOMATIC
srvctl add service -db BOSTON -service SALESRO -role PHYSICAL_STANDBY -policy
AUTOMATIC
```

On `CHICAGO`:

```
srvctl add service -db CHICAGO -s SALESRW -role PRIMARY -y AUTOMATIC
srvctl add service -db CHICAGO -s SALESRO -role PHYSICAL_STANDBY -policy AUTOMATIC
```

To start things up initially, you must manually start the services on the right node. You must also start and stop the `SALESRO` service on the primary so that it can be started on the standby. Issue the following `SRVCTL` commands:

On `BOSTON`:

```
srvctl start service -db BOSTON -service SALESRW
srvctl start service -db BOSTON -service SALESRO
srvctl stop service -db BOSTON -service SALESRO
```

On `CHICAGO`:

```
srvctl start service -db CHICAGO -service SALESRO
```

Now the correct services are running on the correct databases.

If the broker performs a switchover or failover, then it starts the service `SALESRW` or `SALESRO` based on the current role of the database. So `SALESRW` will start on `CHICAGO` (which is now the primary) and `SALESRO` will start on `BOSTON` (which is now the physical standby). The same thing happens if a shutdown and startup of either database occurs - the service that is started is the one that matches the role of the database being started.

Be aware that if you issue the following manual commands on either of those databases, then both the `SALESRO` and `SALESRW` services would be started on the databases regardless of what you may have earlier specified with the `SRVCTL -role` qualifier. This is because the `-role` qualifier is taken into account only by Data Guard broker, and at database startup.

On `BOSTON`:

```
srvctl start service -db CHICAGO
```

On `CHICAGO`:

```
srvctl start service -db BOSTON
```

The `SRVCTL` utility does not automatically take the database role into account, so any time you start a service manually, you must specify the name(s) of the service you want started.

 **Note:**

If the service has been configured to start automatically (`-policy AUTOMATIC`), then the service will automatically start only after a database role change.

 **Note:**

In an Oracle Data Guard configuration, the `SRVCTL -startoption` for a standby database is always set to `OPEN` after a switchover.

 **See Also:**

- For Oracle RAC databases, see *Oracle Real Application Clusters Administration and Deployment Guide* for more information about configuring database services with the `SRVCTL` utility

ONS Configuration Requirements

If client-side ONS configuration is used, the client-side ONS configuration file must specify the hostname and port of the ONS daemon(s) of the primary database and each standby database.

If the client uses remote ONS subscription, the client must specify the hostname and port of the ONS daemon(s) of the primary database and each standby database.

Application Continuity

Application Continuity is an Oracle Database feature that enables rapid and nondisruptive replays of requests against the database after a recoverable error that made the database session unavailable.

Application Continuity is supported for Oracle Data Guard switchovers to physical standby databases. It is also supported for fast-start failover to physical standbys in maximum availability data protection mode. Note that primary and standby databases must be licensed for Oracle RAC or Oracle Active Data Guard in order to use Application Continuity.

 **See Also:**

- *Oracle Real Application Clusters Administration and Deployment Guide* for information about Application Continuity

7

Switchover and Failover in DG PDB Environments

Learn how to use Oracle Data Guard broker to manage switchover and failover operations in a DG PDB environment.

This chapter contains the following topics:

- [Overview of Switchover and Failover in a DG PDB Environment](#)
- [Performing PDB Switchover](#)
- [Performing PDB Failover](#)

Overview of Switchover and Failover in a DG PDB Environment

Oracle Data Guard broker enables you to reverse the roles of the source PDB and its designated target PDB using either a switchover or failover operation.

- PDB switchover

A planned role reversal between a source PDB and its designated target PDB. The broker ensures that the target PDB has applied all the redo generated by the source PDB before switching roles.

As part of managing a switchover, the broker automatically sets up redo transport from the new source PDB to its designated target PDB and starts redo apply services on the new target PDB.

- PDB failover

A role transition in which a designated target PDB is transitioned to the source PDB role after the source PDB has failed or has become unreachable. It is recommended that you perform a failover only when the source PDB is unavailable.

Performing PDB Switchover

A PDB switchover is a role reversal between the source PDB and its designated target PDB. Use DGMGRL commands to perform a PDB switchover.

How Broker Performs Switchover in a DG PDB Environment

The broker performs the following actions when you start a switchover:

1. Verifies that the specified target database is reachable, the target PDB exists, and the target PDB is a designated DG PDB.
2. Validates that the source PDB and target PDB are in the correct states. The source PDB must be open in read/write mode and the target PDB must be in recovery mode.
3. Confirms that the target PDB is recovering the current redo stream of the source PDB.

4. Closes the source PDB on all instances.
5. Converts the source PDB to the standby role. After the switchover completes, the source PDB will become the target PDB.
6. Converts the target PDB to the primary role, after the target PDB has recovered through the all the redo generated by the source PDB and recovery is canceled.
7. Opens the target PDB as the new source PDB.
8. Starts recovery after redo from the target PDB is registered at the original source PDB.

Starting a Switchover to a PDB

When a switchover is started, the source PDB and target PDB must have as small a redo lag as possible.

To start a PDB switchover:

- Run the `SWITCHOVER PLUGGABLE DATABASE` command. Specify the name of the PDB that must now perform the primary role.

The broker controls the rest of the PDB switchover.

Example 7-1 Performing PDB Switchover

This example performs a switchover of the source PDB `sales`. After the operation completes, the target PDB `dgpdb_sales`, in the target database `newyork`, transitions to the primary role and is the new source PDB.

```
DGMGRL> SWITCHOVER TO PLUGGABLE DATABASE 'dgpdb_sales' AT 'newyork';
```

```
Verifying conditions for Switchover...
```

```
Source pluggable database is 'SALES' at database 'BOSTON'
```

```
Performing switchover NOW, please wait...
```

```
Closing pluggable database 'SALES...  
Switching 'SALES' to standby role...  
Waiting for 'DGPDB_SALES' to recover all redo data...  
Stopping recovery at 'DGPDB_SALES'...  
Converting 'DGPDB_SALES' to primary role...  
Opening new primary 'DGPDB_SALES'...  
Waiting for redo data from new primary 'DGPDB_SALES'...  
Starting recovery at new standby 'SALES'...
```

```
Switchover succeeded, new primary is "DGPDB_SALES"
```

Performing PDB Failover

A PDB failover is a role reversal between a source PDB and its designated target PDB. This operation is typically performed when the source PDB has failed and there is no possibility of recovering it in a timely manner.

You can perform one of the following types of PDB failover:

- Complete PDB failover

This is the default. It results in no data loss and is the recommended option. Use this option when there are problems with the source PDB, but the source CDB is still available.

- Immediate PDB failover

This is the fastest type of PDB failover. However, no additional data is applied on the target PDB once you invoke the PDB failover. After an immediate failover operation, you must reinstate the former source PDB before it can serve as a target PDB for the new source PDB. Use this option when the source CDB is not available. There is a high possibility of data loss under these circumstances.

After an immediate failover operation, the former source PDB must be dropped. The new source PDB after the immediate PDB failover is unprotected. A new DGPDB should be added to resume data protection.

Before You Begin DG PDB Failover

Some factors must be considered before you perform a PDB failover.

- Determine that there is no possibility of recovering the source PDB.

How Broker Performs Failover in a DG PDB Environment

The broker performs a set of steps when you initiate a PDB failover.

1. Verifies that the target database is reachable. If the target database is not reachable, you will not be able to perform a failover.
2. Validates that the source PDB and target PDB are in the correct states. The source PDB can be open or closed. The target PDB must be in recovery mode.

Note that this validation is not performed for immediate PDB failover.

3. Confirms that the target PDB is recovering the current redo stream of the source PDB.
4. Closes the source PDB on all instances.
5. Waits for the target database to finish applying any unapplied redo data and then stops Redo Apply. Any unapplied redo will not be applied when performing an immediate failover.
6. Transitions the target PDB to the source PDB as follows:
 - Changes the role of the target PDB to the source PDB.
 - Opens the new source PDB in read/write mode.

After the failover completes, the broker does not run recovery at the new target PDB (original source PDB). Recovery is not started even if you restart the new target database or run an `ENABLE DATABASE` command. After you address the issue that caused the failure, start recovery by running the `EDIT PLUGGABLE DATABASE` command with the `SET STATE=APPLY-ON` option.

Starting a PDB Failover

Use DGMGRL to perform a complete (recommended) or immediate PDB failover.

The source PDB need not be open when performing a failover.

To perform a PDB failover:

1. Connect to the target database.
2. Run the `FAILOVER PLUGGABLE DATABASE` command, specifying the name of the target PDB and the name of the target database containing the target PDB.

For example:

```
DGMGRL> FAILOVER TO PLUGGABLE DATABASE sales AT newyork;
```

All accumulated redo is applied before the target PDB is changed to a source PDB role.

To perform an immediate failover, include the `IMMEDIATE` keyword in the command. In this case, the source PDB role is changed without applying any accumulated redo data.

After the PDB failover operation completes, use the `EDIT PLUGGABLE DATABASE` command with `SET STATE=APPLY-ON` to start redo apply on the former source PDB if it was a complete failover, and any issues that warranted the failover have been addressed.

8

Scenarios Using the DGMGRL Command-Line Interface

Use these scenarios to help you understand what you need to do to start creating, managing, and using an Oracle Data Guard broker configuration.

Read the information about prerequisites for getting started using the Oracle Data Guard command-line interface (DGMGRL), so that you can prepare your instances. Then read the scenarios to understand how you can use DGMGRL to create, manage, and monitor a broker configuration.

- [Prerequisites for Getting Started](#)
- [Scenario 1: Creating a Configuration](#)
- [Scenario 2: Setting Database Properties](#)
- [Scenario 3: Enabling the Configuration and Databases](#)
- [Scenario 4: Setting the Configuration Protection Mode](#)
- [Scenario 5: Setting up Maximum Availability Mode with a Far Sync Instance](#)
- [Scenario 6: Enabling Fast-Start Failover and Starting the Observer](#)
- [Scenario 7: Enabling Fast-Start Failover When a Far Sync Instance Is In Use](#)
- [Scenario 8: Performing Routine Management Tasks](#)
- [Scenario 9: Performing a Switchover Operation](#)
- [Scenario 10: Performing a Manual Failover Operation](#)
- [Scenario 11: Reinstating a Failed Primary Database](#)
- [Scenario 12: Converting a Physical Standby to a Snapshot Standby](#)
- [Scenario 13: Monitoring a Data Guard Configuration](#)
- [Scenario 14: Adding a Recovery Appliance to a Broker Configuration](#)
- [Scenario 15: Exporting and Importing a Broker Configuration File](#)
- [Scenario 16: Using the Observe-only Mode for Fast-Start Failover](#)

Prerequisites for Getting Started

One of the prerequisites for using DGMGRL is that a primary database and any standby databases must already exist.

The `DG_BROKER_START` initialization parameter must be set to `TRUE` for all databases in the configuration. You must use a server parameter file with the broker.

Convert the initialization parameter files (PFILE) on both primary and standby databases into server parameter files (SPFILE), if necessary. Use the following SQL*Plus command:

```
CREATE SPFILE='spfilename' FROM PFILE='pfilename';
```

If an instance was not started with a server parameter file, then you must shut down the instance and restart it using the server parameter file.

After starting the Oracle instance, set the `DG_BROKER_START=TRUE` initialization parameter using the `SQL ALTER SYSTEM` statement. The parameter value will be saved in the server parameter file. The next time you start the Oracle instance, the broker is started automatically, and you do not need to issue the `SQL ALTER SYSTEM` statement again.

The following assumptions are made in these scenarios:

- TCP/IP is used to connect to primary and standby databases.
- The standby database has been created from backups of the primary database control files and datafiles as described in the *Oracle Data Guard Concepts and Administration*.
- The scenarios assume the following broker configuration:
 - The configuration name is `DRSolution`.
 - The database unique name (`DB_UNIQUE_NAME`) of the primary database is `North_Sales`.
 - The database unique name (`DB_UNIQUE_NAME`) of the remote standby database is `South_Sales`.
 - The protection mode is maximum performance mode.
 - There are standby redo log files configured for both the primary and standby database. The transport mode for both databases is `ASYNC`.
 - The standby database is a physical standby database.

Scenario 1: Creating a Configuration

These examples create a broker configuration named `DRSolution` that includes a primary and standby database named `North_Sales` and `South_Sales`.

To create a configuration and add one physical standby database, perform the following tasks:

- [Creating a Configuration Task 1: Invoke DGMGRL](#)
- [Creating a Configuration Task 2: Connect to the Primary Database](#)
- [Creating a Configuration Task 3: Clear Existing Remote Redo Transport Destinations](#)
- [Creating a Configuration Task 4: Create the Broker Configuration](#)
- [Creating a Configuration Task 5: Show the Configuration Information](#)
- [Creating a Configuration Task 6: Add a Standby Database to the Configuration](#)

Creating a Configuration Task 1: Invoke DGMGRL

To start DGMGRL, enter `dgmgrl` at the command-line prompt on a system where Oracle Data Guard is installed.

```
$ dgmgrl
```

The DGMGRL prompt is displayed:

```
DGMGRL>
```

Creating a Configuration Task 2: Connect to the Primary Database

Before you specify any command (other than the `HELP`, `EXIT`, or `QUIT`), you must first connect to the primary database using the DGMGRL `CONNECT` command.

The account from which you connect to the database (`SYS` in this example) must have `SYSDG` or `SYSDBA` privileges on the primary and standby databases.



Note:

If no `AS` clause is specified on the `CONNECT` command, the connection is made as `SYSDBA`.

The following examples show two variations of the `CONNECT` command. [Example 8-1](#) shows how to connect to the default database on the local system, and [Example 8-2](#) includes the Oracle Net Services connect identifier (`North_Sales.example.com`) to make a connection to a database located on a remote system. In both examples, you are prompted for a password.

Example 8-1 Connecting to the Primary Database on the Local System

```
DGMGRL> CONNECT sysdg;  
Password: password  
Connected to "North_Sales"  
Connected as SYSDG.
```

Example 8-2 Connecting to the Primary Database on a Remote System

```
DGMGRL> CONNECT sysdg@North_Sales.example.com;  
Password: password  
Connected to "North_Sales"  
Connected as SYSDG.
```

Creating a Configuration Task 3: Clear Existing Remote Redo Transport Destinations on Standbys and Far Sync Instances To Be Added.

You must clear any remote redo transport destinations on standby databases and far sync instances before those standbys and far syncs can be added to a configuration.

If the remote redo transport destinations are not cleared, then the following error message is returned when you attempt to create the configuration:

```
ORA-16698: LOG_ARCHIVE_DEST_n parameter set for object to be added  
  
Failed.
```

To clear `LOG_ARCHIVE_DEST_n` settings, use the `ALTER SYSTEM SET LOG_ARCHIVE_DEST_n="" SQL*Plus` command.

Remote redo transport destinations on the primary (whether they have the `REGISTER` or `NOREGISTER` attribute) can be left as is.

Creating a Configuration Task 4: Create the Broker Configuration

A broker configuration is initially created with just a primary database.

In this case, the primary is called `North_Sales`. In a later command, you will add the standby database, `South_Sales`.

Note:

The names for the primary and standby databases must match their database unique names. Fetch these from their `DB_UNIQUE_NAME` initialization parameter as follows:

```
SQL> SHOW PARAMETER DB_UNIQUE_NAME;
```

Use the `CREATE CONFIGURATION` command to create the `DRSolution` configuration and define the primary database, `North_Sales`:

```
DGMGRL> CREATE CONFIGURATION 'DRSolution' AS  
> PRIMARY DATABASE IS 'North_Sales'  
> CONNECT IDENTIFIER IS North_Sales.example.com;
```

DGMGRL returns the following information:

```
Configuration "DRSolution" created with primary database "North_Sales"
```

The name `North_Sales` is the value of this database's `DB_UNIQUE_NAME` initialization parameter.

Creating a Configuration Task 5: Show the Configuration Information

Use the `SHOW CONFIGURATION` command to display a brief summary of the configuration.

```
DGMGRL> SHOW CONFIGURATION;
```

DGMGRL returns the following information:

```
Configuration - DRSolution  
  
Protection Mode: MaxPerformance  
Members:  
  North_Sales - Primary database  
  
Fast-Start Failover: DISABLED  
  
Configuration Status:  
DISABLED
```

Creating a Configuration Task 6: Add a Standby Database to the Configuration

To add a standby database to the `DRSolution` configuration, use the `ADD DATABASE` command.

The following command defines `South_Sales` as a standby database, which is the standby database associated with the primary database called `North_Sales`:

```
DGMGRL> ADD DATABASE 'South_Sales' AS
> CONNECT IDENTIFIER IS South_Sales.example.com;
```

DGMGRL returns the following information:

```
Database "South_Sales" added
```

The name `South_Sales` is the value of the database's `DB_UNIQUE_NAME` initialization parameter.

Use the `SHOW CONFIGURATION` command to verify that the `South_Sales` database was added to the `DRSolution` configuration:

```
DGMGRL> SHOW CONFIGURATION;
```

DGMGRL returns the following information:

```
Configuration - DRSolution

Protection Mode: MaxPerformance
Members:
  North_Sales - Primary database
  South_Sales - Physical standby database

Fast-Start Failover: DISABLED

Configuration Status:
DISABLED

DGMGRL>
```

Scenario 2: Setting Database Properties

After you create the configuration with DGMGRL, you can set database properties at any time.

For example, the following statement sets the `StandbyArchiveLocation` database property for the `South_Sales` standby database:

```
DGMGRL> EDIT DATABASE 'South_Sales' SET PROPERTY 'StandbyArchiveLocation'='/archfs/arch/';
Property "StandbyArchiveLocation" updated.
```

Use the `SHOW DATABASE VERBOSE` command to view all properties and their values for a database. The following example shows the properties for the `South_Sales` database.

```
DGMGRL> SHOW DATABASE VERBOSE 'South_Sales'
```

Database - South_Sales

```

Role:                PHYSICAL STANDBY
Intended State:      APPLY-ON
Transport Lag:       0 seconds (computed 0 seconds ago)
Apply Lag:           0 seconds (computed 0 seconds ago)
Average Apply Rate:  2.00 KByte/s
Active Apply Rate:   0 Byte/s
Maximum Apply Rate:  0 Byte/s
Real Time Query:    ON
Instance(s):
    South_sales1
  
```

Properties:

```

DGConnectIdentifier      = 'South_Sales.example.com'
ObserverConnectIdentifier = ''
FastStartFailoverTarget  = ''
PreferredObserverHosts   = ''
LogShipping              = 'ON'
RedoRoutes                = ''
LogXptMode               = 'ASYNC'
DelayMins                 = '0'
Binding                   = 'optional'
MaxFailure                = '0'
ReopenSecs               = '300'
NetTimeout                = '30'
RedoCompression          = 'DISABLE'
PreferredApplyInstance    = ''
ApplyInstanceTimeout      = '0'
ApplyLagThreshold         = '0'
TransportLagThreshold     = '0'
TransportDisconnectedThreshold = '0'
ApplyParallel            = 'AUTO'
ApplyInstances            = '0'
ArchiveLocation           = ''
AlternateLocation         = ''
StandbyArchiveLocation    = ''
StandbyAlternateLocation  = ''
InconsistentProperties    = '(monitor)'
InconsistentLogXptProps   = '(monitor)'
LogXptStatus              = '(monitor)'
SendQEntries              = '(monitor)'
RecvQEntries              = '(monitor)'
HostName                  = 'South_Sales.example.com'
StaticConnectIdentifier    = ''
'(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=tcp)
(HOST=South_Sales.example.com) (PORT=2879)))
(CONNECT_DATA=(SERVICE_NAME=South_Sales_DGMGRL.example.com)
(INSTANCE_NAME=south_sales1) (SERVER=DEDICATED)))'
    TopWaitEvents          = '(monitor)'
    SidName                 = '(monitor)'
  
```

Log file locations:

```

Alert log                : /db/oracle/log/diag/rdbms/South_Sales/
south_sales1/trace/alert_south_sales1.log
  
```

```
Data Guard Broker log   : /db/oracle/log/diag/rdbms/South_Sales/  
south_sales1/trace/drcsouth_sales1.log
```

```
Database Status:  
SUCCESS
```

If broker management of the database is enabled, setting a database property value causes the underlying parameter value to be changed in the corresponding database, and the value for the changed parameter is reflected in the server parameter file. Thus, if the database is shut down and restarted outside of Oracle Enterprise Manager Cloud Control (Cloud Control) and DGMGRL (such as from the SQL*Plus interface), the database uses the new parameter values from the updated server parameter file when it starts. However, you should not make changes to the redo transport services initialization parameters through SQL statements. Doing so will cause an inconsistency between the database and the broker.

 **Note:**

The database properties are typically displayed in mixed-case (for example, `LogXptMode`) typeface to help you visually differentiate database properties (from the corresponding initialization parameter, SQL statement, or PL/SQL procedure), which are typically documented in `UPPERCASE` typeface. However, the commands to manage properties are not case sensitive; you can issue commands in uppercase, lowercase, or mixed-case.

You can change a property if the database is enabled or disabled. However, if the database is disabled when you change a property, the change does not take effect until the database is enabled.

Scenario 3: Enabling the Configuration and Databases

So far, the `DRSolution` configuration is disabled, which means it is not under the control of the Data Guard broker.

When you finish configuring the databases into a broker configuration and setting any necessary database properties, you must enable the configuration to allow the Data Guard broker to manage it.

You can enable:

- The entire configuration, including all of its databases
- A standby database

Enable the entire configuration

You can enable the entire configuration, including all of the databases, with the following command:

```
DGMGRL> ENABLE CONFIGURATION;  
Enabled.
```

Show the configuration

Use the `SHOW` command to verify that the configuration and its databases were successfully enabled:

```
DGMGRL> SHOW CONFIGURATION;
```

DGMGRL returns the following information:

```
Configuration - DRSolution

  Protection Mode: MaxPerformance
  Members:
    North_Sales - Primary database
    South_Sales - Physical standby database

Fast-Start Failover: DISABLED

Configuration Status:
SUCCESS
```

Enable the database

This step is unnecessary except if the standby database was previously disabled with the `DISABLE DATABASE` command. Normally, enabling the configuration also enables the standby database.

```
DGMGRL> ENABLE DATABASE 'South_Sales';
Enabled.
```

Show the database

```
DGMGRL> SHOW DATABASE 'South_Sales';

Database - South_Sales

  Role:                PHYSICAL STANDBY
  Intended State:      APPLY-ON
  Transport Lag:       0 seconds (computed 1 second ago)
  Apply Lag:           0 seconds (computed 1 second ago)
  Apply Rate:          1.54 MByte/s
  Real Time Query:    OFF
  Instance(s):
    south_sales1

Database Status:
SUCCESS
```

Scenario 4: Setting the Configuration Protection Mode

You can change the protection mode of the configuration at any time.

 **Note:**

You cannot change the protection mode from maximum performance mode to maximum protection mode. You must first change the protection mode to maximum availability and then to maximum protection mode.

A restart of the primary database is not necessary when changing the protection mode.

This scenario sets the protection mode of the configuration to the `MAXAVAILABILITY` mode. Note that this protection mode requires that there be at least one standby configured to use standby redo log files and it must receive redo via `SYNC` or `FASTSYNC` mode if it receives redo directly from the primary database. If the standby receives redo via a far sync instance, then the far sync instance must receive redo via `SYNC` or `FASTSYNC` mode and the standby must receive redo from the far sync instance via `ASYNCR` mode.

1. Configure standby redo log files, if necessary.

Because you will be setting the protection mode to the `MAXAVAILABILITY` mode, it is important to ensure that sufficient standby redo log files are configured on the standby database. For more information on setting up redo transport, see *Oracle Data Guard Concepts and Administration*.

2. Configure redo transport mode appropriately.

Configure the standby to receive redo via `SYNC` or `FASTSYNC` mode, if the standby receives redo directly from the primary database. If the standby receives primary redo via a far sync instance, then configure the far sync instance to receive redo via `SYNC` or `FASTSYNC` mode and configure the standby to receive redo via `ASYNCR` mode. For example:

```
DGMGRL> EDIT DATABASE 'South_Sales' SET PROPERTY 'LogXptMode'='SYNC';
Property "LogXptMode" updated
```

The broker will not allow this command to succeed unless the standby database is configured with standby redo log files in the configuration.

3. Change the overall protection mode for the configuration.

Use the `EDIT CONFIGURATION` command to upgrade the broker configuration to the `MAXAVAILABILITY` protection mode:

```
DGMGRL> EDIT CONFIGURATION SET PROTECTION MODE AS MAXAVAILABILITY;
Succeeded.
```

If the configuration is disabled when you enter this command, the actual protection mode change is not applied until you enable the configuration with the `ENABLE CONFIGURATION` command. The broker will not allow you to enable the configuration if it does not find a standby database in the configuration that can support the requirements of the protection mode.

4. Verify the protection mode has changed.

Use the `SHOW CONFIGURATION` command to display the current protection mode for the configuration:

```
DGMGRL> SHOW CONFIGURATION;
```

```
Configuration - DRSolution
```

```

Protection Mode: MaxAvailability
Members:
  North_Sales - Primary database
  South_Sales - Physical standby database

Fast-Start Failover: DISABLED

Configuration Status:
SUCCESS

```



See Also:

[Managing Data Protection Modes](#)

Scenario 5: Setting up Maximum Availability Mode with a Far Sync Instance

A far sync instance can be used with maximum availability protection mode if the primary and standby database are geographically far enough apart to make the use of synchronous transport mode impractical.

The example in this topic shows how to add a far sync instance to the configuration and then set up the `RedoRoutes` property for all members of the configuration. Setting of `RedoRoutes` property for the far sync instance enables it to send redo data based on either the `North_Sales` or `South_Sales` database being the primary.

1. Issue the following commands to add the far sync instance named `FS` to the broker configuration:

```

DGMGRL> ADD FAR_SYNC 'FS' AS CONNECT IDENTIFIER IS FS.EXAMPLE.COM;
DGMGRL> ENABLE FAR_SYNC 'FS';
DGMGRL> SHOW CONFIGURATION;

```

```

Configuration - DRSolution

```

```

  Protection Mode: MaxPerformance
  Members:
  North_Sales - Primary database
  South_Sales - Physical standby database
  FS - Far Sync

```

```

Fast-Start Failover: DISABLED

```

```

Configuration Status:
SUCCESS

```

In this output, `South_Sales` and `FS` are indented under `North_Sales`. This indicates that the primary is sending redo data to both `South_Sales` and `FS`.

2. Issue the following commands so that the current primary (`North_Sales`) sends redo data to the far sync instance only, and the far sync instance sends redo data to `South_Sales` when `North_Sales` is a primary:

```
DGMGRL> EDIT DATABASE 'North_Sales' SET PROPERTY RedoRoutes='(LOCAL : FS SYNC)';
DGMGRL> EDIT FAR_SYNC 'FS' SET PROPERTY RedoRoutes='(North_Sales : South_Sales)';
DGMGRL> SHOW CONFIGURATION;
```

```
DGMGRL> SHOW CONFIGURATION;
```

```
Configuration - DRSolution
```

```
Protection Mode: MaxPerformance
Members:
North_Sales - Primary database
  FS - Far Sync
  South_Sales - Physical standby database
```

```
Fast-Start Failover: DISABLED
```

```
Configuration Status:
SUCCESS
```

The indentation scheme in the output above indicates that `North_Sales` sends redo data to `FS` and `FS` sends redo data to `South_Sales`.

3. Issue the following commands to upgrade the protection mode to maximum availability:

```
DGMGRL> EDIT CONFIGURATION SET PROTECTION MODE AS MaxAvailability;
DGMGRL> SHOW CONFIGURATION;
```

```
Configuration - DRSolution
```

```
Protection Mode: MaxAvailability
Members:
North_Sales - Primary database
  FS - Far Sync
  South_Sales - Physical standby database
```

```
Fast-Start Failover: DISABLED
```

```
Configuration Status:
SUCCESS
```

If `South_Sales` ever becomes the primary and `North_Sales` becomes a standby database that receives redo data from the far sync instance, `FS`, then the `RedoRoutes` property must be set for `South_Sales`. Additionally, a new rule must be included in the `RedoRoutes` property for `FS` to allow it to send redo data to `North_Sales` when `South_Sales` is the primary database.

If these rules are not created, then `North_Sales` will not be able to receive redo data. To check whether the proper rules have been set, use the `SHOW CONFIGURATION WHEN PRIMARY IS` command to see what the redo transport configuration would be if `South_Sales` were the primary database. For example:

```
DGMGRL> SHOW CONFIGURATION WHEN PRIMARY IS 'South_Sales';
```

```
Configuration when South_Sales is primary - DRSolution
```

```
Members:
South_Sales - Primary database
  FS - Far Sync
  North_Sales - Physical standby database
```

```
Members Not Receiving Redo:
```

```
North_FS - Physical standby database
Warning: ORA-16685: database does not receive redo data
```

To correct this error, set the `RedoRoutes` property for `South_Sales` and `FS` as follows:

```
DGMGRL> EDIT DATABASE 'South_Sales' SET PROPERTY RedoRoutes='(LOCAL : FS
SYNC)';
DGMGRL> EDIT FAR_SYNC 'FS' SET PROPERTY RedoRoutes=('North_Sales :
South_Sales)(South_Sales : North_Sales)';
```

After the change to the `RedoRoutes` property for `South_Sales` and `FS` is complete, use the `SHOW CONFIGURATION WHEN PRIMARY IS` command to confirm that the error has been cleared:

```
DGMGRL> SHOW CONFIGURATION WHEN PRIMARY IS 'South_Sales';

Configuration when South_Sales is primary - DRSolution

Members:
South_Sales - Primary database
FS          - Far Sync
North_Sales - Physical standby database
```

Scenario 6: Enabling Fast-Start Failover and Starting the Observer

You can enable fast-start failover from any site, including the observer site, while connected to any database in the broker configuration.

Enabling fast-start failover does not trigger a failover. Instead, it allows the observer that is monitoring the configuration to initiate a fast-start failover if conditions warrant a failover. This section describes the steps to enable fast-start failover and start the observer where the configuration protection mode is set to maximum availability mode.

1. Ensure standby redo logs are configured on the primary and target standby databases. You must stop log apply services prior to configuring standby redo logs.

See Also:

Oracle Data Guard Concepts and Administration for instructions on configuring standby redo log files

2. Configure the primary and standby databases to receive redo via `SYNC` or `FASTSYNC` or `ASYNC` mode, if they receive redo directly rather than via a far sync instance. If either database receives redo via a far sync instance, then configure the far sync instance to receive redo via `SYNC` or `FASTSYNC` mode and configure the database to receive redo via `ASYNC` mode. For example:

```
DGMGRL> EDIT DATABASE 'North_Sales' SET PROPERTY 'LogXptMode'='SYNC';
Property "LogXptMode" updated

DGMGRL> EDIT DATABASE 'South_Sales' SET PROPERTY 'LogXptMode'='SYNC';
Property "LogXptMode" updated
```

The broker does not allow these commands to succeed unless the databases are configured with standby redo log files.

3. If you have two or more standby databases, set up the `FastStartFailoverTarget` configuration property on the primary database to indicate the desired target standby database. The broker reciprocally sets this property for the target standby database to indicate the primary database as its future target standby database when fast-start failover is actually enabled. There is no need for you set this property on the target standby as this is done for you automatically. For example:

```
DGMGRL> EDIT DATABASE 'North_Sales' SET PROPERTY
FastStartFailoverTarget='South_Sales';
Property "FastStartFailoverTarget" updated
```

4. If it is necessary to upgrade the protection mode, use the following DGMGRL `EDIT CONFIGURATION` command. For example:

```
DGMGRL> EDIT CONFIGURATION SET PROTECTION MODE AS MAXAVAILABILITY;
```

5. If it is not already enabled on the primary and standby databases, enable Flashback Database by issuing the following statements on each database (This examples assumes the database is already in archive log mode):

```
ALTER SYSTEM SET DB_FLASHBACK_RETENTION_TARGET=4320 SCOPE=BOTH;
ALTER DATABASE ARCHIVELOG;
ALTER SYSTEM SET db_recovery_file_dest_size=<size>;
ALTER SYSTEM SET db_recovery_file_dest=<directory-specification>;
ALTER DATABASE FLASHBACK ON;
```

Ensure that the `DB_FLASHBACK_RETENTION_TARGET` initialization parameter is set to a sufficiently large value so that reinstatement is still possible after a prolonged outage.

6. Start up to four observers by logging into the observer computers and running DGMGRL. Connect to the configuration as a user who has the `SYSDG` or `SYSDBA` privilege and then issue the `START OBSERVER` command. Note that the command does not return; that is you will not get the DGMGRL prompt after issuing the command. Make sure Redo Apply has been stopped before enabling flashback database.

```
DGMGRL> CONNECT sysdg@North_Sales.example.com;
Password: password
Connected to "North_Sales"
Connected as SYSDG.
```

```
DGMGRL> START OBSERVER observer1 IN BACKGROUND
> FILE IS /net/sales/dat/oracle/broker/fsfo.dat
> LOGFILE IS /net/sales/dat/oracle/broker/observer.log
> CONNECT IDENTIFIER IS North_Sales
Submitted command "START OBSERVER" using connect identifier "North_Sales"
```

Oracle recommends using this command format for security reasons; no credentials are visible. This practice prevents other users on the system from using a utility (for example, the UNIX `ps` utility) to display connection credentials. It also prevents clear-text passwords from being visible on the user's terminal.

When starting the observer from a script, Oracle recommends that you use a method that supports `'connect /'`, so that database connection credentials do not have to be embedded within the script. If you choose to use a client-side Oracle Wallet as a secure external password store (see *Oracle Database Enterprise User Security Administrator's Guide*), be sure to add credentials for both the primary and fast-start failover target

standby databases. The database connect string that you specify when adding the credentials for each database must match the `ObserverConnectIdentifier` or `DGConnectIdentifier` database property.

When multiple observers are started, one observer is the master observer after fast-start failover is enabled, and the remaining observers are backup observers. Only the master observer can coordinate fast-start failover with Data Guard broker. If the primary and target standby databases stay connected but the connection to the master observer is lost, then the broker tries to nominate a backup observer as the new master observer.

7. Enable fast-start failover. You can enable fast-start failover while connected to any database system in the broker configuration. For example:

```
DGMGRL> ENABLE FAST_START FAILOVER;
Enabled in Zero Data Loss Mode.
```

8. Verify the fast-start failover configuration. Use the `SHOW FAST_START FAILOVER` command to display the fast-start failover settings:

```
DGMGRL> SHOW FAST_START FAILOVER;
```

```
Fast-Start Failover: DISABLED

Protection Mode: MaxAvailability
Lag Limit: 0 seconds

Threshold: 30 seconds
Ping Interval: 3000 milliseconds
Ping Retry: 0
Active Target: (none)
Potential Targets: "South_Sales"
  South_Sales valid
Observer: observer.example.com
Shutdown Primary: TRUE
Auto-reinstate: TRUE
Observer Reconnect: (none)
Observer Override: FALSE
```

```
Configurable Failover Conditions
```

```
Health Conditions:
Corrupted Controlfile  YES
Corrupted Dictionary  YES
Inaccessible Logfile   NO
Stuck Archiver         NO
Datafile Write Errors  YES
```

```
Oracle Error Conditions:
(none)
```

The following commands show that the `FastStartFailoverTarget` property is set up reciprocally once fast-start failover is enabled. The first command, issued for the current primary database `North_Sales`, shows the value of the `FastStartFailoverTarget` property to be the current target standby, `South_Sales`. The second command, issued for the target standby `South_Sales`, shows the current primary, `North_Sales`, as the target standby's future target standby should it ever take over as a primary.

```
DGMGRL> SHOW DATABASE 'North_Sales' FastStartFailoverTarget;
FastStartFailoverTarget='South_Sales';
```

```
DGMGRL> SHOW DATABASE 'South_Sales' FastStartFailoverTarget;
FastStartFailoverTarget='North_Sales';
```

Scenario 7: Enabling Fast-Start Failover When a Far Sync Instance Is In Use

Fast-start failover can be enabled in maximum availability mode when the fast-start failover target is a logical or physical standby database that receives redo data from a far sync instance.

To enable fast-start failover when a far sync instance is used to ship redo data to the standby database, the `FastStartFailoverTarget` property must first be set on both the primary and target standby database, as follows:

```
DGMGRL> EDIT DATABASE 'North_Sales' SET PROPERTY FastStartFailoverTarget='South_Sales';
DGMGRL> EDIT DATABASE 'South_Sales' SET PROPERTY FastStartFailoverTarget='North_Sales';
```

Then, fast-start failover can be enabled, as follows:

```
DGMGRL> ENABLE FAST_START FAILOVER;
```

Note that the far sync instance database is not specified as the fast-start failover target for either `North_Sales` or `South_Sales`.

Scenario 8: Performing Routine Management Tasks

There may be situations in which you want to change the state or properties of the databases in a broker configuration to perform routine maintenance on one or more databases.

You might also need to temporarily disable broker management of databases in a configuration.

Changing Properties and States

As you monitor the configuration, you might need to dynamically modify the states of the databases or their properties.

The following topics show how to change the state or properties of the databases in the configuration.

Alter a Database Property

You can modify the values of database properties at any time—whether the database is enabled or disabled.

The following example shows how to use the `EDIT DATABASE` command to change the `LogXptMode` database property to the value `ASYNC` for the `North_Sales` database.

```
DGMGRL> EDIT DATABASE 'North_Sales' SET PROPERTY 'LogXptMode'=ASYNC;
```

DGMGRL returns the following message to indicate that the `LogXptMode` property was updated successfully in the Data Guard configuration file:

```
Property "LogXptMode" updated
```

If the configuration is currently disabled, the database does not use the new property value until you enable the broker configuration with the `ENABLE CONFIGURATION` command.

Reset a Property to Its Default Value

You can reset a configuration or configurable property to its default value at any time whether the database or configuration is enabled or disabled.

The following example shows how to use the `EDIT DATABASE` command to reset the `LogXptMode` database configurable property to its default value for the `North_Sales` database.

```
EDIT DATABASE 'North_Sales' RESET PROPERTY LogXptMode;
```

The following example shows how to use the `EDIT CONFIGURATION` command to reset the `TraceLevel` configuration property to its default value.

```
EDIT CONFIGURATION RESET PROPERTY TraceLevel;
```

Alter the State of a Standby Database

You can temporarily stop Redo Apply on a physical standby.

To change the state of the standby database to `APPLY-OFF`, enter the `EDIT DATABASE` command as shown in the following example.

```
DGMGRL> EDIT DATABASE 'South_Sales' SET STATE='APPLY-OFF';  
Succeeded.
```

Redo data is still being received when you put the physical standby database in the `APPLY-OFF` state.

Alter the State of a Primary Database

You can stop the transmittal of redo data to the standby database.

To change the state of the primary database to accommodate this, use the following command:

```
DGMGRL> EDIT DATABASE North_Sales SET STATE=TRANSPORT-OFF;  
  
Succeeded.
```

To change the state of the primary database back to `TRANSPORT-ON`, do the following:

```
DGMGRL> EDIT DATABASE North_Sales SET STATE=TRANSPORT-ON;  
  
Succeeded.
```

Disabling the Configuration and Databases

When you disable the broker configuration or any of its databases, you are disabling the broker's management of them and are effectively removing your ability to use `DGMGRL` to manage and monitor them.

However, disabling the broker's management of a broker configuration does not affect the actual operation of the underlying Oracle Data Guard configuration or the databases. For example, the redo transport services and log apply services in the Oracle Data Guard configuration continue to function unchanged, but you can no longer manage them.

Disable a Configuration

You must use the `DISABLE CONFIGURATION` command to disable management of the entire broker configuration including the primary database.

For example:

```
DGMGRL> DISABLE CONFIGURATION;
```

The only way to disable broker management of the primary database is to use the `DISABLE CONFIGURATION` command; the `DISABLE DATABASE` command only disables management of a standby database. Likewise, the `DISABLE FAR_SYNC` command only disables management of a far sync instance.

Note:

If you disable management of a configuration while connected to the standby database or far sync instance, you must connect to the primary database (that is, a database whose control file role is primary) to reenab the configuration.

Disabling the broker's management of a configuration member does not remove the member from the broker configuration file. You can reenab your ability to use DGMGRL (or Cloud Control) to manage the member by entering the appropriate `ENABLE CONFIGURATION` or `ENABLE DATABASE` command.

Disable a Standby Database

You use the `DISABLE DATABASE` command when you temporarily do not want the broker to manage and monitor a standby database.

You can explicitly disable broker management of a standby database to prevent it from being enabled when the rest of the configuration is enabled. The following example shows how to disable the `South_Sales` standby database.

```
DGMGRL> DISABLE DATABASE 'South_Sales';  
Disabled.
```

Note:

You cannot disable a standby database from the configuration if fast-start failover is enabled and the database to be disabled is the target standby database.

 **Note:**

If you disable management of a standby database while connected to that standby database, you must connect to the primary database or another enabled standby database to reenable broker-management of the standby database.

 **WARNING:**

If you disable broker management of a standby database in the broker configuration, that standby database cannot be used by the broker as a failover target in the event of loss of the primary database.

When operating under either maximum protection mode or maximum availability mode, the broker prevents you from disabling the last standby database that supports the protection mode.

Disabling a Far Sync Instance

Use the `DISABLE FAR_SYNC` command when you temporarily do not want the broker to manage and monitor a far sync instance.

You can explicitly disable broker management of a far sync instance to prevent it from being enabled when the rest of the configuration is enabled. The following example shows how to disable the far sync instance.

```
DGMGRL> DISABLE FAR_SYNC 'FS';  
Disabled.
```

 **Note:**

The following restrictions apply when disabling a far sync instance:

- You cannot disable a far sync instance if it is specified in the `RedoRoutes` property of any other configuration member.
- If you disable management of a far sync instance while connected to that far sync instance, you must connect to the primary database or another enabled standby database to reenable broker management of the far sync instance.

 **Caution:**

If you disable broker management of a far sync instance in the broker configuration, that far sync instance cannot be specified in a `RedoRoutes` property for any other configuration member.

Removing the Configuration, a Standby Database, or a Far Sync Instance

When you use the `REMOVE CONFIGURATION`, `REMOVE DATABASE`, or `REMOVE FAR_SYNC` command, you effectively delete the configuration, standby database, or far sync instance from the broker configuration file, removing the ability of Oracle Data Guard broker to manage them.

A remove operation with the `PRESERVE DESTINATIONS` clause *does not* remove or delete the actual Oracle Data Guard configuration underneath, nor does it affect the operation of the actual Oracle Data Guard configuration and its databases.

Note:

After you use the `REMOVE CONFIGURATION`, `REMOVE DATABASE`, or `REMOVE FAR_SYNC` command, you must reissue the command(s) that you originally issued if you decide to re-create the deleted object. You must go through the steps in [Scenario 1: Creating a Configuration](#) as necessary, to create a broker configuration that can be managed with DGMGRL (or Cloud Control).

Note:

The following restrictions apply:

- You cannot remove a standby database from the configuration if fast-start failover is enabled and the database to be removed is the target standby database.
- You cannot remove a standby database or a far sync instance if it is specified in the `RedoRoutes` property for any other member in the configuration.

Removing a Standby Database from the Configuration

When you use the `REMOVE DATABASE` command, broker management and monitoring of the database ceases and the database is deleted from the broker configuration file.

Show the configuration before deletion of the `South_Sales` standby database:

```
DGMGRL> SHOW CONFIGURATION;

Configuration - DRSolution

  Protection Mode: MaxPerformance
  Members:
    North_Sales  - Primary database
    FS           - Far Sync
    South_Sales  - Physical standby database

Fast-Start Failover: DISABLED

Configuration Status:
SUCCESS
```

Issue the `DGMGRL REMOVE DATABASE` command to remove the `South_Sales` database information from the Data Guard configuration file:

```
DGMGRL> REMOVE DATABASE 'South_Sales';  
Removed database "South_Sales" from the configuration
```

Show the configuration after deletion of the `South_Sales` standby database:

```
DGMGRL> SHOW CONFIGURATION;  
  
Configuration - DRSolution  
  
Protection Mode: MaxPerformance  
Members:  
North_Sales - Primary database  
FS - Far Sync  
  
Fast-Start Failover: DISABLED  
  
Configuration Status:  
SUCCESS
```

When operating under either maximum protection mode or maximum availability mode, the broker prevents you from deleting the last standby database that supports the protection mode.

Removing a Far Sync Instance from the Configuration

Use the `REMOVE FAR_SYNC` command to remove far sync instance information from the Oracle Data Guard configuration file.

For example, use the following command to remove the `FS` far sync instance information:

```
DGMGRL> REMOVE FAR_SYNC 'FS';  
Removed far sync instance "FS" from the configuration
```

Show the configuration after deletion of the `FS` far sync instance:

```
DGMGRL> SHOW CONFIGURATION;  
  
Configuration - DRSolution  
  
Protection Mode: MaxPerformance  
Members:  
North_Sales - Primary database  
  
Fast-Start Failover: DISABLED  
  
Configuration Status:  
SUCCESS
```

Removing a Broker Configuration

Use the `DGMGRL REMOVE CONFIGURATION` command to remove the entire configuration from management and monitoring by the broker.

For example:

```
DGMGRL> REMOVE CONFIGURATION;  
Removed configuration
```

**Note:**

You cannot remove the configuration if fast-start failover is enabled.

DGMGRL returns the following message to indicate the command successfully removed all of the configuration information from the Data Guard configuration file:

```
DGMGRL> SHOW CONFIGURATION;  
Error: ORA-16532: Data Guard broker configuration does not exist
```

Configuration details cannot be determined by DGMGRL

Scenario 9: Performing a Switchover Operation

You can switch the role of the primary database and a standby database using the `SWITCHOVER` command.

Before you issue the `SWITCHOVER` command, you must ensure:

- The state of the primary and standby databases are `TRANSPORT-ON` and `APPLY-ON`, respectively.
- All participating databases are in good health, without any errors or warnings present.
- The standby database properties were set on the primary database, so that the primary database can function correctly when transitioning to a standby database (shown in the following examples in boldface type).
- Standby redo log files are configured on the primary database.
- If the configuration is in maximum availability mode, then the current primary is configured to receive redo via `SYNC` or `FASTSYNC` or `ASYNCR` mode if it will receive redo directly from the new primary. If it will receive redo via a far sync instance, then the far sync instance is configured to receive redo via `SYNC` or `FASTSYNC` mode and the current primary is configured to receive redo via `ASYNCR` mode. If the configuration is in maximum protection mode, then the current primary is configured to receive redo via `SYNC` mode.
- If fast-start failover is enabled, you can perform a switchover only to the standby database that was specified as the target standby database.

The following are the tasks necessary to perform a switchover using the `SWITCHOVER` command:

- [Using the SWITCHOVER Command Task 1: Check the Primary Database](#)
- [Using the SWITCHOVER Command Task 2: Check the Standby Database That is the Target of the Switchover](#)
- [Using the SWITCHOVER Command Task 3: Confirm That the Database Is Ready for a Role Change](#)
- [Using the SWITCHOVER Command Task 4: Issue the Switchover Command](#)
- [Using the SWITCHOVER Command Task 5: Show the Configuration](#)

Using the SWITCHOVER Command Task 1: Check the Primary Database

Use the `SHOW DATABASE VERBOSE` command to check the state, health, and properties of the primary database.

For example:

```
SHOW DATABASE VERBOSE 'North_Sales';
```

```
Database - North_Sales
```

```
Role:                PRIMARY
Intended State:      TRANSPORT-ON
Instance(s):
  north_sales1
```

```
Properties:
```

```
DGConnectIdentifier      = 'North_Sales.example.com'
ObserverConnectIdentifier = ''
FastStartFailoverTarget  = ''
PreferredObserverHosts   = ''
LogShipping              = 'ON'
RedoRoutes               = ''
LogXptMode               = 'ASYNC'
DelayMins                = '0'
Binding                  = 'optional'
MaxFailure               = '0'
ReopenSecs              = '300'
NetTimeout               = '30'
RedoCompression         = 'DISABLE'
PreferredApplyInstance   = ''
ApplyInstanceTimeout     = '0'
ApplyLagThreshold        = '0'
TransportLagThreshold    = '0'
TransportDisconnectedThreshold = '0'
ApplyParallel           = 'AUTO'
ApplyInstances           = '0'
ArchiveLocation          = ''
AlternateLocation        = ''
StandbyArchiveLocation   = ''
StandbyAlternateLocation = ''
InconsistentProperties   = '(monitor)'
InconsistentLogXptProps  = '(monitor)'
LogXptStatus            = '(monitor)'
SendQEntries            = '(monitor)'
RecvQEntries            = '(monitor)'
HostName                 = 'North_Sales.example.com'
StaticConnectIdentifier  =
'(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)
(HOST=North_Sales.example.com)(PORT=2840))
(CONNECT_DATA=(SERVICE_NAME=North_Sales_DGMGRL.example.com)
(INSTANCE_NAME=north_sales1)(SERVER=DEDICATED)))'
```

```
TopWaitEvents          = '(monitor)'  
SidName                = '(monitor)'  
  
Log file locations:  
Alert log              : /dev/oracle/log/diag/rdbms/North_Sales/  
north_sales1/trace/alert_north_sales1.log  
Data Guard Broker log  : /dev/oracle/log/diag/rdbms/North_Sales/  
north_sales1/trace/drcnorth_sales1.log  
  
Database Status:  
SUCCESS
```

In particular, you should examine the boldface properties and the current status of the primary database.

Using the SWITCHOVER Command Task 2: Check the Standby Database That is the Target of the Switchover

Use the `SHOW DATABASE` command to check the status of the standby database that is the target of the switchover.

For example:

```
DGMGRL> SHOW DATABASE 'South_Sales';  
  
Database - South_Sales  
  
Role: PHYSICAL STANDBY  
Intended State: APPLY-ON  
Transport Lag: 0 seconds (computed 0 seconds ago)  
Apply Lag: 0 seconds (computed 0 seconds ago)  
Apply Rate: 1.44 MByte/s  
Real Time Query: OFF  
Instance(s):  
    south_sales1  
  
Database Status:  
SUCCESS
```

Using the SWITCHOVER Command Task 3: Confirm That the Database Is Ready for a Role Change

Prior to performing a role change, you can use the `VALIDATE DATABASE` command to perform an exhaustive set of checks on the database to confirm that it is ready for a role change.

The examples shown in this step use the `VALIDATE DATABASE` command for all three databases in the `DRSolution` configuration: a primary, logical standby, and physical standby database. The configuration looks as follows:

```
DGMGRL> SHOW CONFIGURATION;  
  
Configuration - DRSolution
```

```

Protection Mode: MaxAvailability
Members:
North_Sales - Primary database
West_Sales - Logical standby database
South_Sales - Physical standby database

Fast-Start Failover: DISABLED

Configuration Status:
SUCCESS

```

Example: Validate the Primary Database

```

DGMGRL> VALIDATE DATABASE North_Sales;

Database Role:      Primary database

Ready for Switchover:  Yes

Managed by Clusterware:
North_Sales:  YES

```

Example: Validate the Logical Standby Database

Validate the logical standby database, as follows:

```

DGMGRL> VALIDATE DATABASE West_Sales;

Database Role:      Logical standby database
Primary Database:  North_Sales

Ready for Switchover:  Yes
Ready for Failover:   Yes (Primary Running)
Warning: Physical and Snapshot standby databases will be
be disabled if a role change is performed to this database

Managed by Clusterware:
North_Sales :  NO
West_Sales  :  NO
The static connect identifier allows for a connection to database
"North_Sales".

Parameter Settings:
Parameter                               North_Sales Value
South_Sales Value
DB_BLOCK_CHECKING                        true                true
DB_BLOCK_CHECKSUM                        true                true
DB_LOST_WRITE_PROTECT                    NONE                NONE

```

Example: Validate the Physical Standby Database

Validate the physical standby database, as follows:

```
DGMGRL> VALIDATE DATABASE South_Sales;

Database Role:      Physical standby database
Primary Database:  North_Sales

Ready for Switchover:  Yes
Ready for Failover:   Yes (Primary Running)

Managed by Clusterware:
  North_Sales : NO
  South_Sales : NO
  The static connect identifier allows for a connection to database
  "North_Sales".

Parameter Settings:
Parameter                                     North_Sales Value      South_Sales
Value
DB_BLOCK_CHECKING                             true                    true
DB_BLOCK_CHECKSUM                             true                    true
DB_LOST_WRITE_PROTECT                         NONE                    NONE
```

Because the configuration protection is set to maximum availability mode and database `South_Sales` is the only one that has its `LogXptMode` property set to `SYNC`, an error is displayed to indicate that a switchover is not possible. Note that when a database receives redo from a database or a far sync instance that has the `RedoRoutes` property configured with a transport mode, that mode overrides the transport mode specified by `LogXptMode`.

Using the SWITCHOVER Command Task 4: Issue the Switchover Command

Issue the `SWITCHOVER` command to swap the roles of the primary and standby databases.

The following example shows how the broker automatically shuts down and restarts the old primary database as a part of the switchover. (See the usage notes in [DGMGRL Command Usage Notes](#) for information about how to set up the broker environment so that DGMGRL can automatically restart the primary and standby databases for you.)

```
DGMGRL> SWITCHOVER TO 'South_Sales';
Performing switchover NOW, please wait...
Operation requires a connection to instance "south_sales1" on database "South_Sales"
Connecting to instance "south_sales1"...
Connected as SYSDBA.
New primary database "South_Sales" is opening...
Operation requires startup of instance "north_sales1" on database "North_Sales"
Starting instance "north_sales1"...
ORACLE instance started.
Database mounted.
Switchover succeeded, new primary is "South_Sales"
```

After the switchover completes, use the `SHOW CONFIGURATION` and `SHOW DATABASE` commands to verify that the switchover operation was successful.

Using the SWITCHOVER Command Task 5: Show the Configuration

Use the `SHOW CONFIGURATION` command to verify that the switchover was successful.

```
DGMGRL> SHOW CONFIGURATION;

Configuration - DRSolution

Protection Mode: MaxAvailability
Members:
  South_Sales - Primary database
  West_Sales - Logical standby database
  North_Sales - Physical standby database

Fast-Start Failover: DISABLED

Configuration Status:
SUCCESS
```

Scenario 10: Performing a Manual Failover Operation

You invoke a failover operation in response to an emergency situation, usually when the primary database cannot be accessed or is unavailable.

See [Choosing a Target Standby Database](#) before you fail over to decide which standby database should be the target of the failover. The following scenario describes a failover to the remote database called `South_Sales`.

Note:

If multiple fast-start failover targets are configured, then a manual failover is only possible to the current fast-start failover target.

If you want to perform a manual failover to a standby database that is not the fast-start failover target standby database, you must first disable fast-start failover using the `FORCE` option on the standby database you want to fail over.

See [Disabling Fast-Start Failover](#) for more information about the `FORCE` option.

1. (Optional) Check the readiness of the target standby.

To validate the target standby database to ensure that it's ready to become the new primary database, use the `VALIDATE DATABASE` command, as shown in the following example:

```
DGMGRL> VALIDATE DATABASE 'South_Sales';

Database Role: Physical standby database
Primary Database: South_Sales

Ready for Switchover: Yes
```

Ready for Failover: Yes

Managed by Clusterware:

North_Sales : NO

South_Sales: NO

The static connect identifier allows for a connection to database "North_Sales".

Parameter Settings:

Parameter Value	North_Sales Value	South_Sales
DB_BLOCK_CHECKING	true	true
DB_BLOCK_CHECKSUM	true	true
DB_LOST_WRITE_PROTECT	NONE	NONE

- To perform the failover operation, you must connect to the standby database to which you want to fail over as a user that has the SYSDBG or SYSDBA privilege. For example:

```
DGMGRL> CONNECT sysdbg@South_Sales.example.com;
Password: password
Connected to "South_Sales"
Connected as SYSDBG.
```

- Now you can issue the failover command to make the target standby database the new primary database for the configuration.

```
DGMGRL> FAILOVER TO 'South_Sales';
Performing failover NOW, please wait...
Failover succeeded, new primary is "South_Sales"
```

Note that after the failover completes, the original primary database cannot be used as a standby database of the new primary database unless it is reinstated or re-created (as described in [Reenabling Disabled Databases After a Role Change](#)).

- Issue the SHOW CONFIGURATION command to verify the failover.

```
DGMGRL> SHOW CONFIGURATION;

Configuration - DRSolution

Protection Mode: MaxAvailability
Members:
  South_Sales - Primary database
    Warning: ORA-16629: database reports a different protection level
    from the protection mode

  North_Sales - Physical standby database (disabled)
    ORA-16661: the standby database needs to be reinstated

Fast-Start Failover: DISABLED

Configuration Status:
WARNING
```

Note that in this example, the configuration was operating in maximum availability mode. The protection mode was preserved after the failover. The configuration also has a warning status. The `SHOW DATABASE` command for the new primary shows that the warning is the result of not having an enabled physical standby database. As a result, the warning status indicates that the protection level of the configuration is not the same as the configured mode.

5. Show the new primary database.

```
DGMGRL> SHOW DATABASE South_Sales;
Database - South_Sales

Role:          PRIMARY
Intended State: TRANSPORT-ON
Instance(s):
  south_sales1

Database Warning(s):
  ORA-16629: database reports a different protection level from
the protection mode

Database Status:
WARNING
```

6. Issue the `SHOW DATABASE` command to see that the former (failed) primary database was disabled by the broker as a consequence of the failover. It must be reinstated (as described in [Reenabling Disabled Databases After a Role Change](#)).

```
DGMGRL> SHOW DATABASE 'North_Sales';
Database - North_Sales

Role: PHYSICAL STANDBY
Intended State: APPLY-ON
Transport Lag: (unknown)
Apply Lag: (unknown)
Apply Rate: (unknown)
Real Time Query: OFF
Instance(s):
  north_sales1

Database Status:
ORA-16661: the standby database needs to be reinstated
```

Scenario 11: Reinstating a Failed Primary Database

If your former primary database was configured with Flashback Database, you can easily reinstate the failed primary database as a standby database of the new primary database.

The failed primary database will be reinstated as a standby type that matches the old standby database. For example, if you failed over to a physical standby database, the old primary will be reinstated as a physical standby database.

To reinstate the failed primary database, start it to the mounted state. Then run DGMGRL, connect to the new primary database and reinstate the old primary database.

1. Restart the old primary database:

```
% dgmgrl connect sysdg
Password: password
Connected to "North_Sales"
Connected as SYSDBG.
```

```
DGMGRL> startup mount;
ORACLE instance started.
Database mounted.
```

2. Reinststate the old primary database:

```
% dgmgrl connect sysdg
Password: password
Connected to "North_Sales"
Connected as SYSDBG.
```

```
DGMGRL> REINSTATE DATABASE 'North_Sales';
Reinstating database "North_Sales", please wait...
Reinstatement of database "North_Sales" succeeded
```

```
Database dismounted.
ORACLE instance shut down.
Operation requires startup of instance "north_sales1" on database "North_Sales"
Starting instance "north_sales1"...
ORACLE instance started.
Database mounted.
Continuing to reinstate database "North_Sales" ...
Reinstatement of database "North_Sales" succeeded
```

After the primary has been reinstated, issue the `SHOW CONFIGURATION` and `SHOW DATABASE` commands to confirm that the old primary has been successfully reinstated.

3. Show the configuration and its members:

```
DGMGRL> SHOW CONFIGURATION
```

```
Configuration - DRSolution

Protection Mode: MaxAvailability
Members:
  South_Sales - Primary database
  North_Sales - Physical standby database
```

```
Fast-Start Failover: DISABLED
```

```
Configuration Status:
SUCCESS
```

```
DGMGRL> SHOW DATABASE 'South_Sales';
```

```
Database - South_Sales

Role:          PRIMARY
Intended State: TRANSPORT-ON
Instance(s):
  south_sales1
```

```

Database Status:
SUCCESS

DGMGRL> SHOW DATABASE 'North_Sales'

Database - 'North_Sales'

Role: PHYSICAL STANDBY
Intended State: APPLY-ON
Transport Lag: 0 seconds (computed 1 second ago)
Apply Lag: 0 seconds (computed 1 second ago)
Apply Rate: 0 Byte/s
Real Time Query: OFF
Instance(s):
  north_sales1

Database Status:
SUCCESS

```

Scenario 12: Converting a Physical Standby to a Snapshot Standby

If you have a physical standby database that you would like to convert to a snapshot standby database, use the `DGMGRL CONVERT DATABASE` command.

Redo data will continue to be received by the database while it is operating as a snapshot standby database, but it will not be applied until the snapshot standby is converted back into a physical standby database.

A physical standby database must be configured with a fast recovery area to convert it to a snapshot standby database. This is because a guaranteed restore point is created during the conversion process, and guaranteed restore points require a fast recovery area.

```

DGMGRL> convert database 'South_Sales' to snapshot standby;
Converting database "South_Sales" to a Snapshot Standby database, please wait...
Database "South_Sales" converted successfully

```

```

DGMGRL> SHOW CONFIGURATION;

```

```

Configuration - DRSolution

Protection Mode: MaxPerformance
Members:
  North_Sales - Primary database
  South_Sales - Snapshot standby database

```

```

Fast-Start Failover: DISABLED

```

```

Configuration Status:
SUCCESS

```

When you are ready to revert the database back to a physical standby database, use the `DGMGRL CONVERT DATABASE` command again as follows. Any updates made to the database while it was operating as a snapshot standby database will be discarded. All accumulated redo data will be applied by Redo Apply services after the database is converted back to a physical standby database.

```
DGMGRL> CONVERT DATABASE 'South_Sales' to PHYSICAL STANDBY;

Converting database "South_Sales" to a Physical Standby database, please wait...
Operation requires shutdown of instance "south_sales1" on database "South_Sales"
Shutting down instance "south_sales1"...
Database closed.
Database dismounted.
ORACLE instance shut down.
Operation requires startup of instance "south_sales1" on database "South_Sales"
Starting instance "south_sales1"...
ORACLE instance started.
Database mounted.
Continuing to convert database "South_Sales" ...
Database "South_Sales" converted successfully
```

Scenario 13: Monitoring a Data Guard Configuration

These steps demonstrate the tasks necessary to use the `SHOW` command and monitorable properties to identify and resolve a failure situation.

- [Monitoring a Configuration Task 1: Check the Configuration Status](#)
- [Monitoring a Configuration Task 2: Check the Database Status](#)
- [Monitoring a Configuration Task 3: Check the LogXptStatus Monitorable Property](#)
- [Monitoring a Configuration Task 4: Check the InconsistentLogXptProps Monitorable Property](#)

Monitoring a Configuration Task 1: Check the Configuration Status

The status of the broker configuration is an aggregated status of all databases and instances in the broker configuration.

You can check the configuration status first to determine whether or not any further action needs to be taken. If the configuration status is `SUCCESS`, everything in the broker configuration is working properly. However, if you see a status of `WARNING` or `ERROR`, then something is wrong in the configuration.

For example, in the following display, you can see that the primary database has multiple warnings:

```
DGMGRL> SHOW CONFIGURATION;
Configuration - DRSolution

  Protection Mode: MaxPerformance
  Members:
    North_Sales - Primary database
      Warning: ORA-16809: multiple warnings detected for the database

    South_Sales - Physical standby database

Fast-Start Failover: DISABLED

Configuration Status:
WARNING
```

Monitoring a Configuration Task 2: Check the Database Status

To identify the warnings on the primary database, show its status using the `SHOW DATABASE` command.

For example:

```
DGMGRL> SHOW DATABASE 'North_Sales';
Database - North_Sales

Role:                PRIMARY
Intended State:      TRANSPORT-ON
Instance(s):
  north_sales1
  Warning: ORA-16737: the redo transport service for standby "South_Sales"
has an error
  Warning: ORA-16714: the value of property LogArchiveTrace is inconsistent
with the database setting
  Warning: ORA-16715: redo transport-related property ReopenSecs of standby
database "South_Sales" is inconsistent

Database Status:
WARNING
```

Monitoring a Configuration Task 3: Check the LogXptStatus Monitorable Property

The `SHOW DATABASE` output in step 2 shows a Warning for error ORA-16737.

To identify the exact transport error, use the `LogXptStatus` monitorable property:

```
DGMGRL> SHOW DATABASE 'North_Sales' 'LogXptStatus';
LOG TRANSPORT STATUS
PRIMARY_INSTANCE_NAME  STANDBY_DATABASE_NAME  STATUS
      north_sales1           South_Sales  ORA-12541: TNS:no listener
```

The output shows that the listener for the physical standby database is not running. To fix this error, start the listener for the physical standby database `South_Sales`.

Monitoring a Configuration Task 4: Check the InconsistentLogXptProps Monitorable Property

To identify the inconsistent values for the redo transport database property, `ReopenSecs`, you can use the `InconsistentLogXptProps` monitorable property.

This is useful, for example, for the warning shown in the `SHOW DATABASE` display in Step 2 is ORA-16715.

```
DGMGRL> SHOW DATABASE 'North_Sales' 'InconsistentLogXptProps';
INCONSISTENT LOG TRANSPORT PROPERTIES
INSTANCE_NAME  STANDBY_NAME  PROPERTY_NAME  MEMORY_VALUE  BROKER_VALUE
  south_sales1  South_Sales  ReopenSecs      600           300
```

The current database memory value (600) is different from the Oracle Data Guard broker's property value (300). If you think the broker's property value is correct, you

can fix the inconsistency by re-editing the property of the standby database with the same value, as shown in the following example:

```
DGMGRL> EDIT DATABASE 'South_Sales' SET PROPERTY 'ReopenSecs'=300;
Property "ReopenSecs" updated
```

You can also reenable the standby database or reset the state of the primary database to `TRANSPORT-ON` to fix this inconsistency.

Scenario 14: Adding a Recovery Appliance to a Broker Configuration

These steps show how to add a Zero Data Loss Recovery Appliance (Recovery Appliance) to a broker configuration.

See Also:

[Example 4-9](#) for an example of how to set up a Recovery Appliance as the redo destination of a physical standby

1. Add the Recovery Appliance to the broker configuration.

```
DGMGRL> ADD RECOVERY_APPLIANCE EnterpriseRecoveryAppliance AS CONNECT IDENTIFIER IS
EnterpriseRecoveryAppliance.example.com;
Oracle Recovery Appliance "EnterpriseRecoveryAppliance" added
```

```
DGMGRL> SHOW RECOVERY_APPLIANCE 'EnterpriseRecoveryAppliance';
Oracle Recovery Appliance - EnterpriseRecoveryAppliance
Transport Lag: 0 seconds
Redo Source: North_Sales
```

```
Oracle Recovery Appliance Status:
DISABLED
```

2. Enable the Recovery Appliance

```
DGMGRL> ENABLE RECOVERY_APPLIANCE 'EnterpriseRecoveryAppliance';
```

```
DGMGRL> SHOW RECOVERY_APPLIANCE 'EnterpriseRecoveryAppliance';
Oracle Recovery Appliance - EnterpriseRecoveryAppliance
Transport Lag: 0 seconds
Redo Source: North_Sales
```

```
Oracle Recovery Appliance Status:
SUCCESS
```

Scenario 15: Exporting and Importing a Broker Configuration File

You can export the broker configuration metadata into a text file. When you need to recreate the broker configuration, you can import the exported metadata into the current broker configuration.

In this example, you save the broker configuration by exporting the configuration metadata into a text file. The configuration is then restored by importing the text file that you had previously exported.

1. [Exporting a Broker Configuration](#)
2. [Importing a Broker Configuration](#)

Exporting a Broker Configuration

Use the `EXPORT CONFIGURATION` command to export the metadata contained in the broker configuration file to a text file.

The directory in which the broker configuration file is stored must be accessible to the Oracle server process.

1. Connect to the primary database.

```
DGMGRL> CONNECT sysdg@North_Sales.example.com;
Password: password
Connected to "North_Sales"
Connected as SYSDBG.
```

2. Export the broker configuration.

The following command exports the broker configuration and stores it in a file named `myconfig.txt` in the trace directory.

```
DGMGRL> EXPORT CONFIGURATION TO 'myconfig.txt';
Succeeded.
```

Importing a Broker Configuration

Use the `IMPORT CONFIGURATION` command to import the broker configuration metadata that is stored in a text file into your current broker configuration.

1. Connect to the primary database.

```
DGMGRL> CONNECT sysdg@North_Sales.example.com;
Password: password
Connected to "North_Sales"
Connected as SYSDBG.
```

2. Import the broker configuration metadata that is stored in the file named `myconfig.txt` in the trace directory into your current broker configuration.

```
DGMGRL> IMPORT CONFIGURATION 'myconfig.txt';
Succeeded.
```

Scenario 16: Using the Observe-only Mode for Fast-Start Failover

The observe-only mode enables you to test the impact of using fast-start failover in your configuration, without making any actual changes to the configuration. You can use the DGMGRL commands or data dictionary views to verify the observe-only mode setting.

Topics:

- [Configuring Observe-only Mode for Fast-Start Failover](#)
- [Sample Content of the Log Files in Observe-only Mode](#)
- [Disabling Observe-only Mode for Fast-start Failover](#)

Configuring Observe-only Mode for Fast-Start Failover

Use the `ENABLE FAST_START FAILOVER` command to configure observe-only mode for fast-start failover.

1. Use the following command to configure the observe-only mode for fast-start failover.

```
DGMGRL> ENABLE FAST_START FAILOVER OBSERVE ONLY;
```

All existing observers and ones that will be started in the future will run in observe-only mode.

2. Verify that the observe-only mode has been set by using one of the following methods:
 - a. Display the current fast-start failover configuration using the following command:

```
DGMGRL> SHOW FAST_START FAILOVER;
Fast-Start Failover: Enabled in Observe-Only Mode
```

```
Protection Mode: MaxAvailability
Lag Limit: 30 seconds
```

```
Threshold: 30 seconds
Ping Interval: 3000 milliseconds
Ping Retry: 0
Active Target: South_Sales
Potential Targets: "South_Sales"
    South_Sales valid
Observer: observer.example.com
Shutdown Primary: TRUE
Auto-reinstate: TRUE
Observer Reconnect: (none)
Observer Override: FALSE
```

- b. Display the current fast-start failover mode using the following command:

```
SQL> SELECT fs_failover_mode from V$DATABASE;
```

```
FS_FAILOVER_MODE
-----
OBSERVE-ONLY
```

Sample Content of the Log Files in Observe-only Mode

This section shows the entries made to the log files when you configure fast-start failover in observe-only mode.

Example 1: When Fast-start Failover Should be Initiated

Observer Log

```
A fast-start failover would have been initiated...
Unable to failover since this observer is in observe-only mode
```

Broker Log

```
Fast-Start Failover cannot proceed because: "observe-only mode"
```

Example 2: Primary Database Opens During Startup Without an Acknowledgement from Observer or Target Standby

The broker log file (drc*.log) and alert log contain the following:

```
This database is allowed to open in observe-only mode. An
acknowledgement from observer or target standby would have been
required in normal FSFO mode.
```

Example 3: Switchover or Manual Failover to a Bystander Database

The broker log file (drc*.log) and alert logs contain the following information:

```
FAILOVER to database 'database name' is allowed even though observe-
only mode is enabled. It would have been rejected since database
'database name' is a bystander database.
```

Disabling Observe-only Mode for Fast-start Failover

Use the `DISABLE FAST_START FAILOVER` command to exit the observe-only mode of fast-start failover. You must first disable fast-start failover and then enable fast-start failover without the `OBSERVE ONLY` clause.

Use the following commands to disable observe-only mode for fast-start failover:

```
DGMGRL> DISABLE FAST_START FAILOVER;
DGMGRL> ENABLE FAST_START FAILOVER;
```

Fast-start failover is now enabled, without observe-only mode.

9

Scenarios for Using DGMGRL with a DG PDB Configuration

Use these scenarios understand how to create, manage, and monitor an Oracle Data Guard broker configuration that provides data protection and disaster recovery for one ore more pluggable databases (PDBs).

Read the information about prerequisites for getting started using the Oracle Data Guard command-line interface (DGMGRL), so that you can prepare your instances. Then read the scenarios to understand how you can use DGMGRL to create, manage, and monitor a broker configuration.

- [Scenario 1: Create the Source and Target Configuration](#)
- [Scenario 2: Establish a Connection Between the Configurations](#)
- [Scenario 3: Setting Up Data Guard for the Source PDB](#)
- [Scenario 4: Enable the DG PDB Configuration and Start Redo Transport](#)
- [Scenario 5: Switchover Source PDB to Target PDB](#)
- [Scenario 6: Failover to Target PDB](#)
- [Scenario 7: Monitoring a DG PDB Configuration](#)

Prerequisites for Using DG PDB

Ensure that the prerequisites for using a DG PDB environment are met.

- The source database and target database must exist.
- You must use a server parameter file. If the source database or target database uses an initialization parameter files (PFILE), convert this file into a server parameter file (SPFILE)
- If an instance was not started with a server parameter file, then you must shut down the instance and restart it using the server parameter file.
- The `DG_BROKER_START` initialization parameter must be set to `TRUE` for the source database and target database in the DG PDB configuration.

The following assumptions are made in the DG PDB scenarios:

- TCP/IP is used to connect to source database with the target database.
- The DG PDB configuration consists of the following:
 - The name of the source configuration is `MyConfig1`. The database unique name (`DB_UNIQUE_NAME`) of the source database is `boston`. The source PDB is named `sales`.
 - The name of the target configuration is `MyConfig2`. The database unique name (`DB_UNIQUE_NAME`) of the target database is `newyork`. The target PDB is named `dgpdb_sales`.
 - The protection mode is maximum performance.

- The transport mode is `ASYN` mode.

Scenario 1: Create the Source and Target Configuration

1. Create the source configuration `MyConfig1`. It must contain the source database `boston` and the PDBs `sales`, `acct`, and `finance`. The source PDB that must be protected using a DG PDB configuration is `sales`.
2. Create the target configuration `MyConfig2`. It must contain the target database `newyork`, which is used to provide data protection to the source PDB `sales` in `MyConfig1`. The source PDB is instantiated as `dgpdb_sales` in target database `newyork`.

See [Scenario 1: Creating a Configuration](#) for information about creating a configuration.

Scenario 2: Establish a Connection Between the Configurations

Establish a connection between the source configuration and the target configuration.

Run the following command, when connected to the primary database `boston` in the configuration `MyConfig1`, to establish a connection with the configuration `MyConfig2`.

```
ADD CONFIGURATION MyConfig2 CONNECT IDENTIFIER IS newyork_ci;
```

Scenario 3: Setting Up Data Guard for the Source PDB

Identify the source PDB to be protected at the target database.

- Use the `ADD PLUGGABLE DATABASE` command to create a DGPDB for the source PDB in the target container database.

```
DGMGRL> ADD PLUGGABLE DATABASE 'dgpdb_sales' AT 'newyork'
SOURCE IS 'sales' AT 'boston'
PDBFileNameConvert IS 'dbs/boston-sales, dbs/newyork-sales-dg';
```

A standby PDB for source PDB `sales` is created as `dgpdb_sales` in the target database `newyork`.

See [ADD PLUGGABLE DATABASE](#) for information about prerequisites and usage information for the `ADD PLUGGABLE DATABASE` command.

Scenario 4: Enable the DG PDB Configuration and Start Redo Transport

Prerequisites

After you add the source PDB, and before you start recovery on the target PDB, ensure that the database files that correspond to the source PDB are copied to the target database. Use RMAN or operating system copy commands to instantiate source PDB files.

The data files that were copied can be from a point in time that is before, during, or after the creation of the target PDB provided they are from the same branch of redo that was active at the source CDB when the target PDB was created.



Note:

When copying the datafiles associated with the source PDB, the naming must follow what's specified in the `PDBFileNameConvert` clause of the `ADD PLUGGABLE DATABASE ...` command.

To enable a DG PDB configuration:

- Use the `ALL` keyword with the `ENABLE CONFIGURATION` command.

```
DGMGRL> ENABLE CONFIGURATION ALL;
Enabled "MyConfig1" with primary database "boston".
Enabled "MyConfig2" with primary database "newyork".
```

- If this is the first DGPDB being added, connect to the target database, set context to the DGPDB, and add the standby redo logs.

Scenario 5: Switchover Source PDB to Target PDB

Use the `SWITCHOVER TO` command switch the role of the source database and the target database in a DG PDB environment.

Prerequisites

- The state of the primary database is `TRANSPORT-ON` and that of the target PDB is `APPLY-ON`.
- The source database and target database are in good health, without any errors or warnings.
- Standby redo log files are configured on the source database.

To perform a switchover for a source PDB:

1. Check the status of the source database and the source PDB.

The following commands displays the state, health, and properties of the source database and source PDB.

```
DGMGRL> SHOW DATABASE boston;

Database - boston
  Role: PRIMARY
  Intended State: TRANSPORT-ON
  PDB Data Guard Role: SOURCE
  Data Guard Source PDB(s): 1
  Instance(s):
    bos1

Database Status:
SUCCESS

DGMGRL> SHOW PLUGGABLE DATABASE 'sales' AT boston;

Pluggable database - SALES at boston

  Data Guard Role: Primary
  Con_ID: 6
  Active Target: con_id 7 at newyork

Pluggable Database Status:
SUCCESS
```

2. Check the status of the target database and target PDB.

The following commands displays the state, health, and properties of the target database and target PDB.

```
DGMGRL> SHOW DATABASE newyork;

Database - newyork
  Role: PRIMARY
  Intended State: TRANSPORT-OFF
  PDB Data Guard Role: TARGET
  Data Guard Target PDB(s): 1
  Instance(s):
    ny1

Database Status:
SUCCESS

DGMGRL> show pluggable database dgpdb_sales at newyork ;

Pluggable database - DGPDB_SALES at newyork

  Data Guard Role: DataGuard Standby
  Con_ID: 7
  Source: con_id 6 at boston
  Transport Lag: 0 seconds (computed 29 seconds ago)
  Intended State: APPLY-ON
  Apply State: Running
  Apply Instance: ny1
```

```
Average Apply Rate: 23 KByte/s  
Real Time Query: OFF
```

```
Pluggable Database Status:  
SUCCESS
```

3. Issue the `SWITCHOVER TO` command to switch to the target database.

The following command switches the source PDB `sales` to its designated target PDB `dgpdb_sales`:

```
DGMGRL> SWITCHOVER TO PLUGGABLE DATABASE dgpdb_sales AT newyork;
```

```
Verifying conditions for Switchover...
```

```
Source pluggable database is 'SALES' at database 'newyork'
```

```
Performing switchover NOW, please wait...
```

```
Closing pluggable database 'SALES'...  
Switching 'SALES' to standby role...  
Waiting for 'DGPDB_SALES' to recover all redo data...  
Stopping recovery at 'DGPDB_SALES'...  
Converting 'DGPDB_SALES' to primary role...  
Opening new primary 'DGPDB_SALES'...  
Waiting for redo data from new primary 'DGPDB_SALES'...  
Starting recovery at new standby 'SALES'...
```

```
Switchover succeeded, new primary is "DGPDB_SALES"  
DGMGRL>
```

4. Verify that the switchover was successful.

The following command displays the DG PDB configuration with source and target databases switched. The individual PDBs show the new roles

```
DGMGRL> SHOW CONFIGURATION;  
Configuration - MyConfig2  
Protection Mode: MaxPerformance  
Members:  
newyork - Primary database  
boston - Primary database at MyConfig1 configuration  
Data Guard Pluggable Database: ENABLED (Source role)  
Configuration Status:  
ENABLED
```

Scenario 6: Failover to Target PDB

You invoke a failover operation in an emergency situation, typically when the source PDB is unavailable or inaccessible. It reverses the roles of the source PDB and its designated target PDB.

In DG PDB environments, if the source PDB is unavailable or inaccessible but the container database and other PDBs are okay, the failover operation is very similar to a switchover

operation in the sense that the original source PDB within the container database is converted and marked as a physical standby, however, recovery will not be started.

To perform a failover:

1. Run the failover command to make the target PDB the new source PDB.

```
DGMGRL> FAILOVER TO PLUGGABLE DATABASE 'dgpdb_sales' AT 'newyork';  
Verifying conditions for Failover...
```

```
Source pluggable database is 'SALES' at database 'newyork'
```

```
Performing failover NOW, please wait...
```

```
Closing pluggable database 'SALES'...  
Switching 'SALES' to standby role...  
Waiting for 'DGPDB_SALES' to recover all redo data...  
Stopping recovery at 'DGPDB_SALES'...  
Converting 'DGPDB_SALES' to primary role...  
Opening new primary 'DGPDB_SALES'...  
Waiting for redo data from new primary 'DGPDB_SALES'...
```

```
Failover succeeded, new primary is "DGPDB_SALES"  
DGMGRL>
```

Note that after the failover completes, recovery is not started on the former (failed) source PDB. It can be used as a target PDB only after it is reinstated.

Note: After the failover completes, recovery is not started on the former (failed) source PDB; not even if the container database is restarted or an `ENABLE DATABASE` is issued at the database level.

2. Verify that the failover operation completed successfully.

```
DGMGRL> SHOW PLUGGABLE DATABASE 'dgpdb_sales' AT newyork;
```

```
Pluggable database 'dgpdb_sales' at database 'newyork'
```

```
Data Guard Role: Primary  
Con_ID: 7  
Active Target: None Available
```

```
Pluggable Database Status:  
ORA-16912: Target pluggable database may have been removed or a  
role change may have occurred.
```

3. Verify that the former (failed) source PDB is not operational as an active standby.

```
DGMGRL> SHOW PLUGGABLE DATABASE sales AT boston;
```

```
Pluggable database 'sales' at database 'boston'  
Data Guard Role: DataGuard Standby  
Con_ID: 7  
Source: (unknown)
```

```
Pluggable Database Status:  
ORA-16911: Redo data from new source database has not been registered.
```

4. Make the former source PDB operational as a target PDB.

After you address the issues that necessitated the failover, you can operationalize the former source PDB so that it functions as a target PDB for the new source PDB.

For example, run the following command to reinstate the former source PDB `sales`:

```
EDIT PLUGGABLE DATABASE sales AT boston SET STATE=APPLY-ON;
```

Scenario 7: Monitoring a DG PDB Configuration

Use DGMGRL to monitor the status of the members in a DG PDB configuration.

Viewing Configuration Information Using DGMGRL

Use the `SHOW CONFIGURATION` command to display the current status of the DG PDB configuration. This command displays the configuration status and roles played by the primary databases in both broker configurations.

```
DGMGRL> SHOW CONFIGURATION;  
Configuration - MyConfig2  
Protection Mode: MaxPerformance  
Members:  
newyork - Primary database  
boston - Primary database at MyConfig1 configuration  
Data Guard Pluggable Database: ENABLED (Target role)  
Configuration Status:  
ENABLED
```

Viewing Database and PDB Information Using DGMGRL

Use the `SHOW PLUGGABLE DATABASE` command to view the status and details of the source database or target database in a DG PDB configuration.

```
DGMGRL> SHOW DATABASE boston;  
  
Database - boston  
  
Role: PRIMARY  
Intended State: TRANSPORT-ON  
DGPluggable State: SOURCE  
Instance(s):  
bos1  
Data Guard Source PDB(s): 1  
  
Database Status:  
SUCCESS
```

Use the `SHOW PLUGGABLE DATABASE` command to view the status of a source PDB or target PDB. Other information includes the role played by the PDB, state, transport lag, and apply lag.

```
DGMGRL> SHOW PLUGGABLE DATABASE dgpdb_sales AT newyork;
Pluggable database - DGPDB_SALES at newyork

Data Guard Role: DataGuard Standby
Con_ID: 7
Source: con_id 6 at boston
Transport Lag: 0 seconds (computed 29 seconds ago)
Intended State: APPLY-ON
Apply State: Running
Apply Instance: nyl
Average Apply Rate: 23 KByte/s
Real Time Query: OFF

Pluggable Database Status:
SUCCESS
```

10

Oracle Data Guard Command-Line Interface Reference

Use the command reference to understand how you can use the Data Guard broker command-line interface (DGMGRL) to manage your broker configuration.

DGMGRL enables you to manage a Data Guard broker configuration and its various members directly from the command line, or from batch programs or scripts. You can use the Data Guard broker command-line interface as an alternative to Oracle Enterprise Manager Cloud Control (Cloud Control) for managing a Data Guard configuration.

- [Starting the Data Guard Command-Line Interface](#)
- [Exiting the Data Guard Command-Line Interface](#)

Note:

A multitenant container database is the only supported architecture in Oracle Database 21c. While the documentation is being revised, legacy terminology may persist. In most cases, "database" and "non-CDB" refer to a CDB or PDB, depending on context. In some contexts, such as upgrades, "non-CDB" refers to a non-CDB from a previous release.

Starting the Data Guard Command-Line Interface

To start the Data Guard command-line interface (DGMGRL), enter `dgmgrl` at the command-line prompt on a system where Oracle is installed.

```
% dgmgrl
```

The DGMGRL command prompt is displayed:

```
DGMGRL>
```

To run DGMGRL, you must have `SYSDG` or `SYSDBA` administrative privilege.

DGMGRL Optional Parameters

You can supply optional parameters on the command line to indicate how you want the Data Guard command-line interface to display output.

Output includes items such as command prompts, banners, and messages.

Additionally, a single command mode is available. In this mode, DGMGRL executes one command and exits upon the completion of the command. The exit code is the result of the command. If the exit code is 0, the command completed successfully. Otherwise, there was an error.

The command line of DGMGRL appears as follows:

```
% dgmgrl [<options>] [<logon> [<command>] ]
```

Specify any of the following keywords when you invoke the DGMGRL command-line interface:

- <options> can be one of the following choices:

- -echo

Displays command input and output to the default display device. If you do not use this parameter, only the output from the command is displayed.

- -logfile <file-spec> "<dgmgrl-command>"

Specifies a file into which you can capture the actions of the DGMGRL command-line interface.

 **Note:**

The DGMGRL `-logfile` option is deprecated as of Oracle Database 12c Release 2 (12.2.0.1). It is supported for backward compatibility only. Instead, the log file should now be specified using the `LOGFILE IS` clause on the `START OBSERVER` command.

 **See Also:**

- * [The "START OBSERVER" command](#)
- * [Capturing Observer Actions in the Observer Log File](#)

- -silent

Suppresses the display of the DGMGRL (`DGMGRL>`) command prompt on your default display device. This option is useful if you are directing the command output to a file or to another display tool.

- <logon> is:

- username [@connect-identifier]

To connect to the database, enter a `username` and optionally, a `connect-identifier`. You will then be prompted for a password. The `connect-identifier` is a fully specified connect descriptor or a name to be resolved by an Oracle naming method (for example, TNS).

If a fully specified connect descriptor is used, it needs to include quotation marks; otherwise the connections will fail with an `invalid option` error. The following is an example of connecting using quotation marks:

```
dgmgrl sys@(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)
(HOST=sales-server)(PORT=1521)))
(CONNECT_DATA=(SERVICE_NAME=sales.us.example.com)))'
```

Enter password: *password*

Whether the connect identifier is specified using a fully specified connect descriptor or using the Easy Connect naming method, any of the following syntax is valid (you will be prompted for a password):

```
* dgmgrl username@'connect_identifier'
* dgmgrl username@"connect_identifier"
* dgmgrl username@'connect_identifier'
```

⚠ WARNING:

Including a password on the command line when invoking DGMGRL is a security risk. This risk can be avoided either by omitting the password when invoking DGMGRL and entering it when prompted, or by using an external authentication method.

- You can connect as `/` when using operating-system authentication (remote database restarts will not work), Secure Sockets Layer (SSL) protocol, or database credentials stored in a wallet.
- `<command>` is a single command.

For example:

```
dgmgrl sys "show database 'North_Sales'"
Password: password
```

The following subsections specify the command format that you enter at the `DGMGRL>` command prompt.

DGMGRL Command Format and Parameters

The DGMGRL commands allow you to create and maintain one broker configuration at a time.

Table 10-1 Summary of DGMGRL Commands

Command	Effect
@ (at sign) Command	Executes a DGMGRL script.
/ (slash) Command	Repeats the last command entered at the DGMGRL command prompt.

Table 10-1 (Cont.) Summary of DGMGRL Commands

Command	Effect
ADD DATABASE	Adds a new standby database to the existing broker configuration.
ADD FAR_SYNC	Adds an existing far sync instance to an Oracle Data Guard broker configuration.
ADD RECOVERY_APPLIANCE	Adds a Zero Data Loss Recovery Appliance (Recovery Appliance) to an existing broker configuration.
CONNECT	Connects to the specified database using the specified username.
CONVERT DATABASE	Converts the specified database to either a snapshot standby database or a physical standby database.
CREATE CONFIGURATION	Creates a broker configuration and adds a primary database to that configuration.
DISABLE CONFIGURATION	Disables broker management of a configuration so that the configuration and all of its databases are no longer managed by the broker.
DISABLE DATABASE	Disables broker management of the named standby database.
DISABLE FAR_SYNC	Disables broker management of a far sync instance.
DISABLE FAST_START FAILOVER	Disables fast-start failover.
DISABLE FAST_START FAILOVER CONDITION	Allows a user to remove conditions for which a fast-start failover should be performed.
DISABLE RECOVERY_APPLIANCE	Disables broker management of the named Zero Data Loss Recovery Appliance (Recovery Appliance).
EDIT CONFIGURATION (Property)	Changes the value of a property for the broker configuration.
EDIT CONFIGURATION (Protection Mode)	Changes the current protection mode setting for the broker configuration.
EDIT CONFIGURATION (RENAME)	Changes the configuration name.
EDIT CONFIGURATION RESET (Property)	Resets the specified configuration property to its default value.
EDIT DATABASE (Property)	Changes the value of a property for the named database.
EDIT DATABASE (Parameter)	Changes the value of a database initialization parameter for the named database.
EDIT DATABASE (Rename)	Changes the name used by the broker to refer to the specified database.
EDIT DATABASE (State)	Changes the state of the specified database.
EDIT DATABASE RESET (Property)	Resets the specified property for the named database to its default value.
EDIT DATABASE RESET (Parameter)	Resets the specified database initialization parameter for the named database.
EDIT FAR_SYNC	Changes the name, state, or properties of a far sync instance.
EDIT FAR_SYNC RESET (Property)	Resets the specified property for the named far sync instance to its default value.
EDIT FAR_SYNC RESET (Parameter)	Resets the specified database initialization parameter for the named far sync instance

Table 10-1 (Cont.) Summary of DGMGRL Commands

Command	Effect
<code>EDIT RECOVERY_APPLIANCE</code> (Property)	Changes the value of the property for the named Zero Data Loss Recovery Appliance (Recovery Appliance).
<code>EDIT RECOVERY_APPLIANCE</code> (Parameter)	Changes the value of the database initialization parameter for the named Zero Data Loss Recovery Appliance (Recovery Appliance).
<code>EDIT RECOVERY_APPLIANCE</code> (Rename)	Changes the name used by the broker to refer to the specified Zero Data Loss Recovery Appliance (Recovery Appliance), as recorded in that Recovery Appliance's profile in the broker configuration.
<code>EDIT RECOVERY_APPLIANCE</code> <code>RESET</code> (Property)	Resets the specified property for the named Zero Data Loss Recovery Appliance (Recovery Appliance) to its default value.
<code>EDIT RECOVERY_APPLIANCE</code> <code>RESET</code> (Parameter)	Resets the specified database initialization parameter for the named Zero Data Loss Recovery Appliance (Recovery Appliance) to its default value.
<code>ENABLE CONFIGURATION</code>	Enables broker management of the broker configuration and all of its databases.
<code>ENABLE DATABASE</code>	Enables broker management of the specified database.
<code>ENABLE FAR_SYNC</code>	Enables broker management of the specified far sync instance.
<code>ENABLE FAST_START FAILOVER</code>	Enables the broker to automatically failover from the primary database to a target standby database.
<code>ENABLE FAST_START FAILOVER</code> <code>CONDITION</code>	Allows a user to add conditions for which a fast-start failover should be performed.
<code>ENABLE RECOVERY_APPLIANCE</code>	Enables broker management of the specified Zero Data Loss Recovery Appliance (Recovery Appliance).
<code>EXIT</code>	Exits the Data Guard command-line interface.
<code>EXPORT CONFIGURATION</code>	Saves the metadata contained in the broker configuration file to a text file.
<code>FAILOVER</code>	Performs a database failover operation in which the standby database, to which DGMGRL is currently connected, fails over to the role of primary database.
<code>HELP</code>	Displays online help for the Data Guard command-line interface.
<code>HOST</code> or <code>!</code> (exclamation point)	Executes operating system command(s) directly through the DGMGRL console without leaving DGMGRL.
<code>IMPORT CONFIGURATION</code>	Import the broker configuration metadata that was previously exported using the <code>EXPORT CONFIGURATION</code> command.
<code>MIGRATE PLUGGABLE DATABASE</code> <code>QUIT</code>	Migrates a PDB from one CDB to another on the same host. Quits the Data Guard command-line interface.
<code>REINSTATE DATABASE</code>	Reinstates the database after a failover.
<code>REMOVE CONFIGURATION</code>	Removes the broker configuration and ends broker management of its members.
<code>REMOVE DATABASE</code>	Removes the specified standby database from the broker configuration.
<code>REMOVE FAR_SYNC</code>	Removes a far sync instance from an Oracle Data Guard broker configuration.

Table 10-1 (Cont.) Summary of DGMGRL Commands

Command	Effect
REMOVE INSTANCE	Removes an instance from the broker configuration.
REMOVE RECOVERY_APPLIANCE	Removes the specified Zero Data Loss Recovery Appliance (Recovery Appliance) from the broker configuration and terminates broker management of the Recovery Appliance.
SET ECHO	Controls whether or not to echo commands that are issued either at the command-line prompt or from a DGMGRL script.
SET FAST_START FAILOVER TARGET	Sets the fast-start failover target to the named standby database.
SET MASTEROBSERVER TO	Lets you manually designate which observer is recognized as the master observer.
SET MASTEROBSERVERHOSTS	Sets the master observer of a broker configuration to the observer on the target host.
SET ObserverConfigFile	Sets the full path and file name of an observer configuration file.
SET TIME	Turns timestamp printing on and off.
SET TRACE_LEVEL	Specifies the amount of tracing done by DGMGRL.
SHOW ALL	Shows the values of DGMGRL CLI properties.
SHOW CONFIGURATION	Displays information about the broker configuration.
SHOW CONFIGURATION WHEN PRIMARY IS	Shows the redo transport configuration that would be in effect if the specified database were the primary database.
SHOW DATABASE	Displays information about the specified database.
SHOW FAR_SYNC	Shows information about a far sync instance.
SHOW FAST_START FAILOVER	Displays all fast-start failover related information.
SHOW INSTANCE	Displays information about the specified instance.
SHOW OBSERVER	Shows information about all registered observers in a Data Guard broker configuration.
SHOW ObserverConfigFile	Shows the value of the ObserverConfigFile property.
SHOW OBSERVERS	Shows information about all observers for all broker configurations in a specific configuration group.
SHOW RECOVERY_APPLIANCE	Displays information or property values of the specified Zero Data Loss Recovery Appliance (Recovery Appliance).
SHUTDOWN	Shuts down a currently running Oracle database.
SPOOL	Records the input and output of DGMGRL to a file.
SQL	Allows you to enter SQL statements from the Data Guard command-line interface (DGMGRL).
START OBSERVER	Starts the observer.
START OBSERVER IN BACKGROUND	Starts a fast-start failover observer as a background process on the host where the DGMGRL session is running.
START OBSERVING	Starts a new observer for each broker configuration in the specified group.
STARTUP	Starts an Oracle instance with the same options as SQL*Plus, including mounting and opening a database.
STOP OBSERVER	Stops the observer.

Table 10-1 (Cont.) Summary of DGMGRL Commands

Command	Effect
<code>STOP OBSERVING</code>	Stops all local observers running on the host where this DGMGRL session is running, for all broker configurations in a specific group.
<code>SWITCHOVER</code>	Performs a switchover operation in which the current primary database becomes a standby database, and the specified standby database becomes the primary database.
<code>VALIDATE DATABASE</code>	Performs a comprehensive set of database checks prior to a role change.
<code>VALIDATE DATABASE DATAFILE</code>	Performs validation of data files across the primary database and standby databases.
<code>VALIDATE DATABASE SPFILE</code>	Performs a comparison of server parameter file (SPFILE) entries between the primary database and a specified standby database.
<code>VALIDATE DGConnectIdentifier</code>	Enables users to check to see whether a connection string is valid for the DGConnectIdentifier property.
<code>VALIDATE FAR_SYNC</code>	Performs a comprehensive set of checks for a far sync instance.
<code>VALIDATE NETWORK CONFIGURATION</code>	Performs network connectivity checks between members of the configuration.
<code>VALIDATE STATIC CONNECT IDENTIFIER</code>	Validates database static connect identifier(s).

DGMGRL Command Usage Notes

The items in this list describe usage notes specific to DGMGRL.

- The `DG_BROKER_START` dynamic initialization parameter is set to `TRUE`.
- To enable broker operations that require restarting instances without manual intervention, Oracle Net Services must be configured on each of the hosts that contain the primary and standby database instances. Specifically, the `listener.ora` file must contain static configuration information about the instance. The `GLOBAL_DBNAME` attribute must be set to `db_unique_name_DGMGRL.db_domain`. See [Prerequisites](#) for additional information.
- The connect identifier used while creating the configuration or adding a database, must be resolvable from any of the hosts in the configuration.
- You must have `SYSDG` or `SYSDBA` privileges to use the Oracle Data Guard command-line interface. Do not include `AS SYSDG` or `AS SYSDBA` on the `CONNECT` command. DGMGRL first attempts an `AS SYSDG` connection; if that fails, it then attempts an `AS SYSDBA` connection.
- If you specify more than one option on the command, you can specify the options in any order.
- A semicolon is required at the end of each DGMGRL command.
- Characters specified in a DGMGRL command string value are interpreted as lowercase characters, unless enclosed in double (") or single (') quotation marks. For example, `database` and `DatAbaSe` are equivalent, but `"database"` and `"DatAbaSe"` are not.

- You can use the backslash (\) to escape a single quotation mark ('), a double quotation mark ("), and the backslash character (\) itself if these characters appear in a character string.
- Some operations on a broker configuration may require that one or more databases be shut down and restarted. In most cases, DGMGRL will automatically shut down and restart a given database for you if the following are true:
 - The *instance-name* is the SID (this applies to Cloud Control as well as DGMGRL).
 - The broker must be able to connect to the database using the same credentials given in the last `CONNECT` command, even if the last `CONNECT` command was used to connect to another database.

Command Examples

Example 10-1 Connecting to a Database Instance on a Local System

This example demonstrates how to connect to a database instance on the local system.

```
% dgmgrl
.
.
.
Welcome to DGMGRL, type "help" for information.

DGMGRL> CONNECT sysdg;
Password: password
Connected to "North_Sales"
Connected as SYSDBG.
```

Example 10-2 Connecting to a Database Instance on a Remote System

This example demonstrates how to connect to a database instance on a remote system.

```
DGMGRL> CONNECT sysdg@remote-stby;
Password: password
Connected to "remote-stdby"
Connected as SYSDBG.
```

Example 10-3 Connecting Using the AS Option

This example demonstrates how to connect to a database instance using the `CONNECT AS` option:

```
DGMGRL> CONNECT sys@remote-stby AS SYSDBA;
Password: password
Connected to "remote-stdby"
Connect as SYSDBA.
```

Exiting the Data Guard Command-Line Interface

When you are done working with the DGMGRL interface and want to return to the operating system, enter the `EXIT` or `QUIT` command.

For example:

```
DGMGRL> EXIT;
```

@ (at sign) Command

The `@` command allows you to execute DGMGRL commands stored in script files.

You can put a sequence of commands into a script file and then use the `@` command to execute the file. The commands contained in the script are executed sequentially.

Format

From within DGMGRL, the syntax is as follows:

```
DGMGRL> @script_file_name
```

Command Parameters

Flag	Description
-echo	Prints all the commands in the script along with their execution results.

Usage Notes

The script that you execute with this command must meet the following qualifications:

- DGMGRL must be able to access the script; otherwise the command fails because DGMGRL cannot open the file.
- Every DGMGRL command included in the script must end with a semi-colon.
- Recursive `@` command execution is allowed, but the limit of recursive levels is 20. If the recursive level reaches 20, then the execution is terminated and none of the unexecuted commands are run. Therefore, self-recursive execution of the `@` command (for example, putting an `@abc.script` command in `abc.script` itself) should be used with caution.
- If there is a `START OBSERVER` command in the script, then any commands that come after it are ignored because the `START OBSERVER` command turns the DGMGRL session into an observer.

The `START OBSERVER IN BACKGROUND` command is treated as a normal command; that is, any commands that come after it are executed.

- Comment lines are permitted in the script, but they must be terminated with a semi-colon. For example the following comments would be allowed in a script:

```
REM Hello World;  
-- Hello Again!;
```

The double dash must be followed by a space character before the comment text.

/ (slash) Command

Use the DGMGRL / (slash) command to repeat the last command entered at the command prompt.

Format

```
DGMGRL> /
```

Usage Notes

- The following commands are not repeatable using the / (slash) command:
 - Return
 - An unrecognized command
 - The `CONNECT` command (because it may contain credentials)
 - The / (slash) command itself

Command Example

In the following example, the / (slash) command is used to easily repeat the `SHOW CONFIGURATION` command.

```
DGMGRL> SHOW CONFIGURATION;
```

```
Configuration - Sales_Configuration
```

```
Protection Mode: MaxAvailability
Members:
North_Sales - Primary database
Local_Sales - Physical standby database
Remote_Sales - Physical standby database (receiving current
redo)
```

```
Fast-Start Failover: DISABLED
```

```
Configuration Status:
SUCCESS
```

```
DGMGRL> /
```

```
Configuration - Sales_Configuration
```

```
Protection Mode: MaxAvailability
Members:
North_Sales - Primary database
Local_Sales - Physical standby database
Remote_Sales - Physical standby database (receiving current
redo)
```

```
Fast-Start Failover: DISABLED
```

```
Configuration Status:  
SUCCESS
```

```
DGMGRL>
```

ADD CONFIGURATION

The `ADD CONFIGURATION` command establishes a relationship between two Data Guard broker configurations by adding a broker configuration to the current configuration.

Prerequisites

- There must be no traditional Data Guard configured by way of standby databases.
- The initialization parameter `DG_BROKER_START` must be set to `TRUE`.
- The `DG_BROKER_CONFIG_FILE` parameter must be configured correctly.
- A server parameter file (spfile) must be created in the source CDB and target CDB.

Format

```
ADD CONFIGURATION config-name CONNECT IDENTIFIER IS dg-connect-identifier;
```

Parameters

Parameter	Description
<code>config-name</code>	Name of the broker configuration that must be added. This configuration must contain one primary database and no standby databases.
<code>dg-connect-identifier</code>	Oracle net connect identifier used to connect to the source database in the configuration <code>config-name</code> . The connect identifier must correspond to a source database.

Usage Notes

- Redo transport is automatically set up between the primary databases in the two broker configurations after the target PDBs are added. However, no primary database is designated as the source or target at this point.
- The configuration name must be different from any other configuration that already exists in the broker metadata.

Examples

Example 10-4 Establishing a Connection Between Two Broker Configurations

This example adds a configuration named `MyConfig2` to the current configuration, `MyConfig1`. The Oracle net connect string to connect to the primary database in `MyConfig2` is `newyork_ci`.

```
DGMGRL> ADD CONFIGURATION 'MyConfig2' CONNECT IDENTIFIER IS newyork_ci;  
Added configuration "MyConfig2" with primary database "newyork".
```

ADD DATABASE

The DGMGRL `ADD DATABASE` command adds a standby database to an existing broker configuration.

Format

```
ADD DATABASE database-name AS CONNECT IDENTIFIER IS connect-  
identifier;
```

Command Parameters

database-name

The name that will be used by the broker to refer to this standby database. It must match (case-insensitive) the value of the corresponding database `DB_UNIQUE_NAME` initialization parameter.

connect-identifier

A fully specified connect descriptor or a name to be resolved by an Oracle Net Services naming method (for example, TNS). The value you specify is also used as the initial value of the `DGConnectIdentifier` database property.

Usage Notes

- To issue this command, you must connect to the primary database or to an enabled standby database that is already in the configuration.
- The broker uses the specified `connect-identifier` to communicate with the specified database from other databases. Therefore, you must ensure that the `connect-identifier` can be used to address the specified database from all databases in your configuration. For example, if TNS is used as the naming method, you must ensure that the `tnsnames.ora` file on every database and instance that is part of the configuration contains an entry for the `connect-identifier`. The connect identifier must resolve to the same connect descriptor. If the database that is being added is an Oracle RAC database, the `connect-identifier` provided here must reach all instances of the Oracle RAC, preferably with `FAILOVER` attributes set.

See Also:

Oracle Database Net Services Administrator's Guide

- If the connection cannot be made, the broker does not add the new database to the configuration.
- You must clear any remote redo transport destinations on the standby database before it can be added to the configuration.

Command Example

The following example shows how to add a database named `South_Sales`.

```
DGMGRL> ADD DATABASE South_Sales AS CONNECT IDENTIFIER IS South_Sales.example.com;
Database "South_Sales" added
```

ADD FAR_SYNC

The `ADD FAR_SYNC` command adds an existing far sync instance to an Oracle Data Guard broker configuration.

The far sync instance is disabled after creation. You must explicitly enable it before the broker can ship redo to and from it.

Format

```
ADD FAR SYNC far_sync_instance_name AS CONNECT IDENTIFIER IS connect-identifier;
```

Command Parameters

far_sync_instance_name

The name that will be used by the broker to refer to this far sync instance. It must match (case-insensitive) the value of the corresponding far sync instance `DB_UNIQUE_NAME` initialization parameter.

connect-identifier

A fully specified connect descriptor or a name to be resolved by an Oracle Net Services naming method (for example, TNS). The value you specify is also used as the initial value of the `DGConnectIdentifier` property.

Usage Notes

- The far sync instance must already exist before you can add it to a broker configuration.
- You must clear any remote redo transport destinations on the far sync instance before it can be added to the configuration.

Command Example

The following example adds a far sync instance named `chicago` to the configuration.

```
DGMGRL> ADD FAR_SYNC chicago AS CONNECT IDENTIFIER IS chicago.example.com;
```

ADD PLUGGABLE DATABASE

This command instantiates a target PDB in the target database. The target PDB is used to provide data protection for a source PDB in a source database.

Format

```
ADD PLUGGABLE DATABASE pdb-name AT TARGET target-database-name  
SOURCE IS source-pdb-name AT source-database-name  
PDBFileNameConvert IS 'file-convert-string';
```

Parameters

Parameter	Description
<code>pdb-name</code>	Name of the target PDB that must be instantiated in the target database. A PDB with the specified name must not exist in the target database.
<code>target-database-name</code>	Name of the target database that contains the target PDB.
<code>source-pdb-name</code>	Name of the source PDB that must be instantiated at the target database. The source PDB must exist in the source database specified by the <code>source-database-name</code> parameter.
<code>source-database-name</code>	Name of the source database that contains the source PDB.
<code>file-convert-string</code>	String conversion of files names from source database to target database.

Usage Notes

- This command instantiates the specified PDB after verifying that it exists in the source database and does not exist in the target database.
- If the specified PDB exists only in one primary database, this database is assumed to be the source database.
- If a PDB with the specified name exists in the target database and is operating as a native PDB (not set up for Data Guard protection), an error is displayed.

Examples

Example 10-5 Instantiating a Source PDB in a DG PDB Environment

This example adds a target PDB named `dgpdb_sales` to the target database named `newyork`. The target PDB is used to provide data protection for the source PDB `sales` in the source database `boston`. The `PDBFileNameConvert` keyword specifies how to convert database files in the source database to the target database.

```
ADD PLUGGABLE DATABASE 'dgpdb_sales' AT TARGET 'newyork'  
SOURCE IS 'sales' AT 'boston'  
PDBFileNameConvert IS 'dbs/boston-sales, dbs/newyork-sales-dg';
```

ADD RECOVERY_APPLIANCE

The `ADD RECOVERY_APPLIANCE` command adds a Zero Data Loss Recovery Appliance (Recovery Appliance) to an existing broker configuration.

The `AS CONNECT IDENTIFIER` clause is optional. If you do not specify this clause, then the broker searches the `LOG_ARCHIVE_DEST_n` initialization parameters on the primary database and all enabled standby databases for an entry that corresponds to the Recovery Appliance being added.

Format

```
ADD RECOVERY APPLIANCE object-name AS CONNECT IDENTIFIER IS connect-identifier;
```

Command Parameters

object name

The name that will be used by the broker to refer to this Recovery Appliance. It must match (case-insensitive) the value of the corresponding Recovery Appliance `DB_UNIQUE_NAME` initialization parameter.

connect-identifier

A fully specified connect descriptor or a name to be resolved by an Oracle Net Services naming method (for example, TNS). The value you specify is also used as the value of the `DGConnectIdentifier` database property.

Usage Notes

- To issue this command, you must connect to the primary database or to an enabled standby database that is already in the configuration.
- The broker uses the specified connect identifier to communicate with the specified Recovery Appliance from any database in the configuration. Therefore, you must ensure that the connect identifier can be used to address the specified Recovery Appliance from any database in the configuration. For example, if TNS is used as the naming method, you must ensure that the `tnsnames.ora` file on every database and instance that is part of the configuration contains an entry for the connect identifier. The connect identifier must resolve to the same connect descriptor.
- If the connection cannot be made, then the broker does not add the new Recovery Appliance to the configuration.
- It is possible to have more than one Recovery Appliance in the configuration.
- Follow the guidelines in this publication to configure Redo Transport services for a Recovery Appliance: [Zero Data Loss Recovery Appliance](#)

Command Example

The following example shows how to add a Recovery Appliance named `EnterpriseRecoveryAppliance`.

```
DGMGRL> ADD RECOVERY_APPLIANCE EnterpriseRecoveryAppliance AS CONNECT IDENTIFIER IS
EnterpriseRecoveryAppliance.example.com;Oracle Backup Appliance
"EnterpriseRecoveryAppliance" added
```

CONNECT

The DGMGRL `CONNECT` command connects you to a database or far sync instance that is a member of a Data Guard broker configuration.

Format

```
CONNECT username@connect-identifier [AS {SYSDBA | SYSDBG}];
```

Command Parameters

username

Represents the username with which you want to connect to the configuration member. You will be prompted for a password after you enter a username and optionally, a connect-identifier.

connect-identifier

This parameter is optional. It is an Oracle Net Services connect identifier for the configuration member to which you want to connect. The exact syntax depends upon the Oracle Net Services communications protocol your Oracle installation uses.

Usage Notes

- The username and password must be valid for the configuration member to which you are trying to connect.
The username you specify must have the `SYSDBG` or `SYSDBA` privilege.
- The `AS` clause is optional. If it is specified, then DGMGRL attempts to connect as either `SYSDBG` or `SYSDBA`, whichever one was specified. If the `AS` clause is *not* specified, then DGMGRL first attempts an `AS SYSDBG` connection; if that fails, it then attempts an `AS SYSDBA` connection.
- Each time the `CONNECT` command is run, the broker checks if the default directories containing the client-side broker files exist.
 - If the `DG_ADMIN` environment variable is defined, and the directory specified in this variable exists with the required permissions, the `log`, `dat`, and `callout` subdirectories are created under the `$DG_ADMIN/config_ConfigurationSimpleName` directory.
 - If the `DG_ADMIN` environment variable is not defined, or the directory specified by `DG_ADMIN` does not have the required permissions, then broker does not create any subdirectories. See [Location of Client-side Broker Files](#) for details about the required permissions.
- If the `CONNECT` command returns an error, check to see that you specified a valid connect-identifier.
- When the `CONNECT` command is successful, the name of the configuration member to which the connection has been made is shown.

Command Examples

Example 1: Connecting to a Local Configuration Member

This example connects to the default database or far sync instance on the local system.

```
DGMGRL> CONNECT sysdg;  
Password: password  
Connected to "North_Sales"  
Connected as SYSDBG.
```

Example 2: Connecting to a Remote Configuration Member

This example connects to a database on the remote system (you could specify a far sync instead).

```
DGMGRL> CONNECT sysdg@South_Sales;  
Password: password  
Connected to "South_Sales"  
Connected as SYSDBG.
```

Example 3: Connecting Without Showing Connection Credentials

This example connects to a configuration member using `CONNECT '/'` so that connection credentials are not visible on the command line:

```
DGMGRL> CONNECT /@North_Sales.example.com;  
Connected to "North_Sales"
```

You must set up Oracle Wallet or SSL to use `CONNECT '/'`. By setting up Oracle Wallet or SSL, you can write a script to securely start and run the observer as a background job without specifying database credentials in the script.

CONVERT DATABASE

The `CONVERT DATABASE` command converts a physical standby database to a snapshot standby database, or reverts the snapshot standby database back to a physical standby database.

A snapshot standby database is a fully updatable standby database. Like a physical or logical standby database, a snapshot standby database receives and archives redo data from a primary database. Unlike a physical or logical standby database, a snapshot standby database does not apply the redo data that it receives. The redo data received by a snapshot standby database is not applied until the snapshot standby is converted back into a physical standby database, after first discarding any local updates made to the snapshot standby database.

A snapshot standby database is best used in scenarios that require a temporary, updatable snapshot of a physical standby database. Note that because redo data received by a snapshot standby database is not applied until it is converted back into a physical standby, the time needed to perform a role transition is directly proportional to the amount of redo data that needs to be applied.

See *Oracle Data Guard Concepts and Administration* for additional information about snapshot standby databases.

Format

```
CONVERT DATABASE db_unique_name TO {SNAPSHOT | PHYSICAL} STANDBY;
```

Usage Notes

- A physical standby database cannot be converted to a snapshot standby database if it is the target of a fast-start failover. The `ORA-16668: operation cannot be performed on the fast-start failover target standby database error will be returned.`
- A physical standby database cannot be converted to a snapshot standby database if its `RedoRoutes` configurable property is set to non-NULL value.
- Use the DGMGRL `ADD DATABASE` command to import an existing snapshot standby database into an Oracle Data Guard broker configuration.
- A snapshot standby database cannot be the target of a switchover or a fast-start failover.
- A snapshot standby database can be the target of a manual failover if fast-start failover is disabled.
- You can use the `SHOW CONFIGURATION` or `SHOW DATABASE` command to verify the conversion result. For example:

```
DGMGRL> SHOW CONFIGURATION;

Configuration - DRSolution

Protection Mode: MaxPerformance
Members:
  North_Sales - Primary database
  South_Sales - Snapshot standby database

Fast-Start Failover: DISABLED

Configuration Status:
SUCCESS
```

- After a snapshot standby database is converted back to a physical standby database, it will be in the default state for a physical standby database, `APPLY-ON`.

Command Examples

Example 1: Converting a Physical Standby to a Snapshot Standby

Issue the following to convert a physical standby database to a snapshot standby database:

```
DGMGRL> CONVERT DATABASE 'South_Sales' to SNAPSHOT STANDBY;
Converting database "South_Sales" to a Snapshot Standby database, please wait...
Database "South_Sales" converted successfully
```

Example 2: Converting a Snapshot Standby Back to a Physical Standby

Issue the following to convert the snapshot standby database back to a physical standby database:

```
DGMGRL> CONVERT DATABASE 'South_Sales' to PHYSICAL STANDBY;
Converting database "South_Sales" to a Physical Standby database, please wait...
Operation requires shutdown of instance "south_sales1" on database "South_Sales"
Shutting down instance "south_sales1"...
Database closed.
Database dismounted.
ORACLE instance shut down.
```

```
Operation requires startup of instance "south_sales1" on database "South_Sales"  
Starting instance "south_sales1"...  
ORACLE instance started.  
Database mounted.  
Continuing to convert database "South_Sales" ...  
Database "South_Sales" converted successfully
```

CREATE CONFIGURATION

The `CREATE CONFIGURATION` command creates a new broker configuration that includes the specified primary database.

Format

```
CREATE CONFIGURATION configuration_name AS PRIMARY DATABASE IS  
database-name CONNECT IDENTIFIER IS connect-identifier;
```

Command Parameters

configuration-name

A user-friendly name for the configuration you are creating. Valid names contain any alphanumeric characters. If spaces are included in the name, the name must be enclosed in double or single quotation marks. The name must consist of 30 or fewer bytes.

database-name

The name that will be used by the broker to refer to the primary database. It must match (case-insensitive) the value of the primary database `DB_UNIQUE_NAME` initialization parameter.

connect-identifier

A fully specified connect descriptor or a name to be resolved by an Oracle Net Services naming method (for example, TNS). The value you specify is also used as the initial value of the `DGConnectIdentifier` database property.

Usage Notes

- A broker configuration is a named collection of one or more databases that you want to manage as a group. You must specify a value for each of the command parameters. There are no default values.
- You must connect to the primary database to issue this command.
- The broker uses the specified `connect-identifier` to communicate with the specified database from other databases. Therefore, you must ensure that the `connect-identifier` can be used to address the specified database from all databases in your configuration. For example, if TNS is used as the naming method, you must ensure that the `tnsnames.ora` file on every database and instance that is part of the configuration contains an entry for the `connect-identifier`. The connect identifier must resolve to the same connect descriptor. If the database that is being added is an Oracle RAC database, the `connect-identifier` provided here must reach all instances of the Oracle RAC, preferably with `FAILOVER` attributes set.

See Also:

Oracle Database Net Services Administrator's Guide

- To add standby databases after you create the broker configuration, use the [ADD DATABASE](#) command.
- You must clear any remote redo transport destinations on the primary database that do not have the `NOREGISTER` attribute, before a configuration can be created.

Command Example

The following example creates a new broker configuration named `DRSolution` with a primary database named `North_Sales`.

```
DGMGRL> CREATE CONFIGURATION 'DRSolution' AS
> PRIMARY DATABASE IS 'North_Sales'
> CONNECT IDENTIFIER IS North_Sales.example.com;
Configuration "DRSolution" created with primary database "North_Sales"
```

CREATE FAR_SYNC

The `CREATE FAR_SYNC` command creates a new far sync instance and adds it to the broker configuration.

Format

```
CREATE FAR_SYNC object-name AS CONNECT IDENTIFIER IS connect-identifier
[ SPFILE [ PARAMETER_VALUE_CONVERT 'string-pair-values' ] [ SET parameter-name value ] ... [ SET parameter-name value ] [ RESET parameter-name ] ... [ RESET parameter-name ] ];
```

Usage Notes

- The version of the connected database must be the same as the DGMGRL version.
- If the `SPFILE` clause is included, the initialization parameters in the specified spfile are used when creating the far sync instance.
This command uses the `RMAN DUPLICATE` command to create a far sync instance. If the auxiliary instance used during `RMAN` duplication was started using an spfile, you cannot include the `SPFILE` clause in the `CREATE FAR_SYNC` command.
- The `PARAMETER_VALUE_CONVERT` clause must immediately follow the `SPFILE` clause. Note that the command fails if you use the `PARAMETER_VALUE_CONVERT` clause after the `SET` or `RESET` clause.
- If the auxiliary instance is started using a parameter file, then broker does the following:
 - Appends the `PARAMETER_VALUE_CONVERT` clause specified in the `CREATE FAR_SYNC` command
 - Sets the following initialization parameters for the far sync instance:
 - * `DB_NAME` to the `DB_NAME` of the primary database
 - * `DB_UNIQUE_NAME` to the value based on user input
 - * `SGA_TARGET` to 300MB
 - * `CPU_COUNT` to 1

- Resets the following initialization parameters: CONTROL_FILES, CLUSTER_DATABASE, DB_RECOVERY_FILE_DEST, DB_RECOVERY_FILE_DEST_SIZE, DB_FILE_NAME_CONVERT, and LOG_ARCHIVE_CONFIG.

Parameters that contribute the total size of SGA memory such as, but not limited to, DB_CACHE_SIZE, LOG_BUFFER, SHARED_POOL_SIZE, LARGE_POOL_SIZE, JAVA_POOL_SIZE, STREAMS_POOL_SIZE are reset.
 - Clears the values of existing LOG_ARCHIVE_DEST_n initialization parameters.
 - Appends the SPFILE clause specified in the CREATE FAR_SYNC command

The values specified in the SPFILE clause take precedence over values specified previously.
 - Appends the NOFILENAMECHECK clause to prevent RMAN from validating the file names during restoring.
- If an initialization parameter of the far sync instance is not set appropriately, the command can fail because DGMGRL fails to start up the far sync instance. In this case, identify the cause of the problem by viewing the alert log files or broker log files and modify the required initialization parameter either in the spfile or by appending the PARAMETER_VALUE_CONVERT, SET, or RESET clause.

Command Parameters

object-name

The value for the DB_UNIQUE_NAME initialization parameter of the far sync instance.

connect-identifier

A fully specified connect descriptor or a name to be resolved by an Oracle Net Services naming method (for example, TNS). The value you specify is also used as the initial value of the DGConnectIdentifier database property.

string-value-pairs

A list of string pairs. The value is set to the value of PARAMETER_VALUE_CONVERT clause when the RMAN DUPLICATE command is invoked.

The PARAMETER_VALUE_CONVERT clause replaces the first string with the second string in all matching initialization parameter values. Multiple pairs of strings may be specified by this parameter.

For example, ' "string1", "string2", "string3", "string4", ... '

Where:

- • string1 is the pattern of initialization parameters of the primary database.
- • string2 is the pattern of initialization parameters of the far sync instance.
- • string3 is the pattern of initialization parameters of the primary database.
- • string4 is the pattern of initialization parameters of the far sync instance.

This is an optional argument. If not specified, a copy of the server parameter file on the primary database will be used on far sync instance without any modification.

parameter-name

Name of the initialization parameter that must be set or reset.

value

Value that must be set for the specified initialization parameter.

Command Example

The following example creates a far sync instance named `bostonfs` and adds it to the broker configuration. The initialization parameters `LOG_FILE_NAME_CONVERT`, `DB_RECOVERY_FILE_DEST`, and `DB_RECOVERY_FILE_DEST_SIZE` are set to the values specified.

```
DGMGRL> CREATE FAR_SYNC bostonfs
        AS CONNECT IDENTIFIER IS "bostonfs_conn_str"
        PARAMETER_VALUE_CONVERT "boston","bostonfs"
        SET LOG_FILE_NAME_CONVERT "boston","bostonfs"
        SET DB_RECOVERY_FILE_DEST "$ORACLE_HOME/dbs/"
        SET DB_RECOVERY_FILE_DEST_SIZE "100G"
        RESET UNDO_TABLESPACE;
```

```
Creating far sync instance "bostonfs".
Connected to "boston"
Connected to "bostonfs"
far sync instance "bostonfs" created
far sync instance "bostonfs" added
```

DISABLE CONFIGURATION

The `DISABLE CONFIGURATION` command disables broker management of a configuration so that the configuration and all of its databases are no longer managed by the broker.

Format

```
DISABLE CONFIGURATION;
```

Command Parameters

None.

Usage Notes

- A disabled configuration and all of its constituent databases are no longer managed by the broker.
- The only way to disable broker management of the primary database is to use the `DISABLE CONFIGURATION` command.
- This command does not remove the broker configuration from the configuration file. See the [REMOVE CONFIGURATION](#) command for more information about removing the configuration.
- You can edit database properties and modify the configuration's protection mode while the configuration is disabled. However, any changes made to properties or to the protection mode will not take effect until the configuration is enabled.
- This command cannot be executed if fast-start failover is enabled.

Command Example

The following example disables management of the broker configuration and all of its databases.

```
DGMGRL> DISABLE CONFIGURATION;  
Disabled.
```

DISABLE DATABASE

The `DISABLE DATABASE` command disables broker management of the named standby database.

This means that broker directed state changes will be disallowed for this database, and the broker will not monitor the database for health status or for monitorable properties.

Format

```
DISABLE DATABASE database-name;
```

Command Parameters

database-name

Name of the standby database to be disabled.

Usage Notes

- You cannot specify the name of a primary database.
- Use the `DISABLE CONFIGURATION` command to disable the primary and all standby databases.
- If the sole standby database is disabled, you have no failover option. This standby database is not viable for failover until it is reenabled.
- This command cannot be used to disable the fast-start failover target database when fast-start failover is enabled.

Command Example

The following example shows how to disable a database named `South_Sales`.

```
DGMGRL> DISABLE DATABASE 'South_Sales';  
Disabled.
```

DISABLE FAR_SYNC

The `DISABLE FAR_SYNC` command disables broker management of a far sync instance.

Format

```
DISABLE FAR_SYNC far_sync_instance_name;
```

Command Parameters

far_sync_instance_name

The name of the far sync instance to be disabled.

Usage Notes

- A far sync instance that has its `RedoRoutes` property set cannot be disabled.

Command Example

The following example disables broker management of a far sync instance named `chicago`.

```
DGMGRL> DISABLE FAR_SYNC 'chicago';
```

DISABLE FAST_START FAILOVER

The `DISABLE FAST_START FAILOVER` command prevents the observer from initiating a failover to the target standby database.

See [Disabling Fast-Start Failover](#) for additional information.

Format

```
DISABLE FAST_START FAILOVER [ FORCE ];
```

Command Parameters

None.

Usage Notes

- If the primary and target standby database have a network connection, use `DISABLE FAST_START FAILOVER` *without* the `FORCE` option to disable fast-start failover on all databases in the broker configuration. If errors occur during the disable operation, the broker returns an error message and stops the disable operation. You may need to reissue the `DISABLE FAST_START FAILOVER` command with the `FORCE` option to override the error conditions and disable fast-start failover on the database to which you are connected. See [Disabling Fast-Start Failover](#) for more information.
- Use `DISABLE FAST_START FAILOVER` *with* the `FORCE` option when the network between the primary and target standby databases is disconnected or when the database upon which the command is received does not have a connection with the primary database. The `FORCE` option disables fast-start failover on the database to which you are connected, even when errors occur.
- Disabling fast-start failover with the `FORCE` option on a primary database that is disconnected from the observer and the target standby database does not prevent the observer from initiating a fast-start failover to the target standby database.
- You can disable fast-start failover while connected to any database in the broker configuration so long as connectivity exists between that database and the primary.
- If disabled by force at the target standby database and the connection subsequently resumes with the primary database, fast-start failover is disabled on all databases in the configuration.
- Disabling fast-start failover with the `FORCE` option while connected to the primary will disable fast-start failover on the target standby database if there is network connectivity between both databases.

Command Examples

Example 1: Disabling a Fast-Start Failover

The following example shows how to disable fast-start failover.

```
DGMGRL> DISABLE FAST_START FAILOVER;  
Disabled.
```

Example 2: Using FORCE When Disabling Fast-Start Failover

The following example uses the `FORCE` option which disables fast-start failover on the database to which you are connected.

```
DGMGRL> DISABLE FAST_START FAILOVER FORCE;  
Disabled.
```

DISABLE FAST_START FAILOVER CONDITION

The `DISABLE FAST_START FAILOVER CONDITION` command allows you to remove conditions for which a fast-start failover should be performed.

Format

```
DISABLE FAST_START FAILOVER CONDITION value;
```

Command Parameters

value

Possible values are any conditions for which a fast-start failover has been enabled.

Usage Notes

If the condition has not been set or if it is an unrecognized condition, then an error is raised.

Command Example

This example specifies that the detection of a corrupted control file does not automatically initiate an immediate fast-start failover.

```
DGMGRL> DISABLE FAST_START FAILOVER CONDITION "Corrupted Controlfile";
```

DISABLE RECOVERY_APPLIANCE

The `DISABLE RECOVERY_APPLIANCE` command disables broker management of the named Zero Data Loss Recovery Appliance (Recovery Appliance).

Disabling broker management of a Recovery Appliance means that the broker will not monitor the health of the transport to the Recovery Appliance. However, redo transport to the Recovery Appliance will not be shut off.

Format

```
DISABLE RECOVERY_APPLIANCE object_name;
```

Command Parameters

object name

Name of the Recovery Appliance to be disabled.

Command Example

The following example shows how to disable a Recovery Appliance named `EnterpriseRecoveryAppliance`.

```
DGMGRL> DISABLE RECOVERY_APPLIANCE 'EnterpriseRecoveryAppliance';  
Disabled.
```

EDIT CONFIGURATION (Property)

The `EDIT CONFIGURATION SET PROPERTY` command changes the value of a property for the broker configuration.

Format

```
EDIT CONFIGURATION SET PROPERTY property-name=value;
```

Command Parameters

property-name

The name of a configuration property.

value

The new value for the property.



See Also:

[Managing the Members of a Broker Configuration and Oracle Data Guard Broker Properties](#) for information about configuration properties

Usage Notes

- Issue this command while connected to the primary database or to any standby database in the broker configuration having connectivity to the primary database.
- Use the `SHOW CONFIGURATION` command to display the current property information for the configuration.

Command Example

The following example shows how to set the `FastStartFailoverThreshold` configuration property to 90 seconds.

```
DGMGRL> EDIT CONFIGURATION SET PROPERTY FastStartFailoverThreshold=90;
```

EDIT CONFIGURATION (Protection Mode)

The `EDIT CONFIGURATION SET PROTECTION MODE AS` command edits the current protection mode setting for the broker configuration.

Format

```
EDIT CONFIGURATION SET PROTECTION MODE AS protection-mode;
```

Command Parameters

protection-mode

The data protection mode in which you want the configuration to run when the configuration is enabled. The possible protection modes are:

```
MAXPROTECTION
MAXAVAILABILITY
MAXPERFORMANCE
```

Usage Notes

- Before you use the `EDIT CONFIGURATION` command to set the protection mode, ensure that at least one standby is configured to receive redo via `SYNC` or `FASTSYNC` mode if it receives redo directly from the primary. If the standby receives redo via a far sync instance, the far sync instance must be configured to receive redo via `SYNC` or `FASTSYNC` mode and the standby must be configured to receive redo via `ASYN` mode.
- The following table shows the configuration protection modes and the minimum corresponding settings for redo transport services:

Protection Mode	Redo Transport	Standby Redo Log Files Needed?	Usable with Fast-Start Failover?
MAXPROTECTION	SYNC	Yes	Yes
MAXAVAILABILITY	SYNC or FASTSYNC	Yes	Yes
MAXPERFORMANCE	ASYN	Yes	Yes

The default protection mode for the configuration is `MAXPERFORMANCE`.

See Also:

[Managing the Members of a Broker Configuration](#) for more information about the protection modes and redo transport services

- This command cannot be executed if fast-start failover is enabled.
- Upgrading from `MAXPERFORMANCE` to `MAXPROTECTION` is not allowed. You must first go to `MAXAVAILABILITY` and then to `MAXPROTECTION`.
- Use the `SHOW CONFIGURATION` command to display the current protection mode for the configuration.

```
DGMGRL> SHOW CONFIGURATION;
```

```
Configuration - DRSolution
```

```
Protection Mode: MaxPerformance
```

```
Members:
```

```
North_Sales - Primary database
```

```
South_Sales - Physical standby database
```

```
Fast-Start Failover: DISABLED
```

```
Configuration Status:  
SUCCESS
```

If broker management of the configuration is disabled when you enter the `EDIT CONFIGURATION` command, the protection mode of the configuration does not take effect until the next time you enable the configuration with the `ENABLE CONFIGURATION` command.

Command Example

The following example shows how to upgrade the broker configuration to the `MAXAVAILABILITY` protection mode.

Verify that standby redo log files are configured on the standby database and that the redo transport service is set to `SYNC`, for example:

```
DGMGRL> EDIT DATABASE 'South_Sales' SET PROPERTY 'LogXptMode'='SYNC';  
Property "LogXptMode" updated
```

```
DGMGRL> EDIT CONFIGURATION SET PROTECTION MODE AS MAXAVAILABILITY;  
Succeeded.
```

EDIT CONFIGURATION (RENAME)

The `EDIT CONFIGURATION RENAME TO` command changes a configuration's name.

Format

```
EDIT CONFIGURATION RENAME TO new-configuration-name;
```

Command Parameters

new-configuration-name

The new name for the configuration.

Command Example

The following example shows how to rename a configuration named `DR_Sales` to `HA_Sales`.

```
DGMGRL> SHOW CONFIGURATION
```

```
Configuration - DR_Sales
```

```
Protection Mode: MaxPerformance
```

```
Members:
```

```
North_Sales - Primary database
```

```
South_Sales - Physical standby database
```

```
Fast-Start Failover: DISABLED
```

```
Configuration Status:  
DISABLED
```

```
DGMGRL> EDIT CONFIGURATION RENAME TO "HA_Sales";
Succeeded.
DGMGRL> ENABLE CONFIGURATION
Enabled.
DGMGRL> SHOW CONFIGURATION

Configuration - HA_Sales

Protection Mode: MaxPerformance
Members:
  North_Sales - Primary database
  South_Sales - Physical standby database

Fast-Start Failover: DISABLED

Configuration Status:
SUCCESS
```

EDIT CONFIGURATION RESET (Property)

The `EDIT CONFIGURATION RESET PROPERTY` command resets the specified configuration property to its default value.

Format

```
EDIT CONFIGURATION RESET PROPERTY property-name;
```

Command Parameters

property-name

The name of an existing configuration property.

Usage Notes

- Issue this command while connected to the primary database or to any standby database in the broker configuration having connectivity to the primary database.
- Use the `SHOW CONFIGURATION` command to display the current property information for the configuration.

Command Example

The following example shows how to reset the `BystandersFollowChange` property.

```
DGMGRL> EDIT CONFIGURATION RESET PROPERTY BystandersFollowChange;
Succeeded.
```

EDIT CONFIGURATION (Set Source)

This command designates the source configuration in a DG PDB environment. Once the target PDB is created, the broker automatically sets up redo transport between the source configuration and target configuration. However, to proactively start redo transport between the configurations, you can use the `EDIT CONFIGURATION` command.

Format

```
EDIT CONFIGURATION config-name SET AS SOURCE [TO target-config-name];
```

Parameters

Parameter	Description
config-name	Name of the source configuration in a DG PDB configuration. The source configuration is the source of the redo data.
target-config-name	Name of the target configuration which receives redo data from the source configuration specified by config-name. Specifying a target configuration name is optional.

Usage Notes

- Use the optional `TO` clause to restrict redo transport only to the primary database within the specified target configuration.
- Once the source configuration is set, the broker starts redo transport services on the source database.

Examples**Example 10-6 Setting the Source Configuration**

This example configures the broker configuration named `MyConfig1` as the source configuration.

```
DGMGRL> EDIT CONFIGURATION MyConfig1 SET AS SOURCE;
Primary database "boston" is redo source for primary database
"newyork".
```

EDIT DATABASE (Property)

The `EDIT DATABASE` command changes the name used by the broker for the specified database.

Format

```
EDIT DATABASE database-name SET PROPERTY property-name=value;
```

Command Parameters**database-name**

The name of the database for which you want to change a property value.

property-name

The name of an existing database-specific property. If this is an Oracle RAC database, this property change affects all instances of the database.

 **See Also:**

[Managing the Members of a Broker Configuration](#) and [Oracle Data Guard Broker Properties](#) for information about properties.

value

The new value for the property.

 **Note:**

This command can be used to change the value of an instance-specific property if and only if just one instance is known by the broker for the named database. An attempt to use this command to change an instance-specific property when the broker knows of multiple instances of the database will be rejected. It is recommended to only use `EDIT INSTANCE (property)` to change the value of an instance-specific property.

Command Examples**Example 1: Editing a Configurable Property at the Database Level**

The following example edits a configurable property at the database level.

```
DGMGRL> EDIT DATABASE 'North_Sales' SET PROPERTY 'LogXptMode'='SYNC';  
Property "LogXptMode" updated
```

Example 2: Editing a Configurable Instance-Specific Property of a non-Oracle RAC Database

The following example edits a configurable instance-specific property of a non-Oracle RAC database.

```
DGMGRL> EDIT DATABASE 'South_Sales' SET PROPERTY  
> 'MaxFailure'=255;  
Property "MaxFailure" updated
```

Example 3: Editing a Configurable Instance-Specific Property of an Oracle RAC Database

The following example edits a configurable instance-specific property of an Oracle RAC database. This will not succeed because it is not clear to which instance the property change should be applied.

```
DGMGRL> EDIT DATABASE 'North_Sales' SET PROPERTY  
> 'MaxFailure'=10;  
Error: ORA-16587: ambiguous object specified to Data Guard broker Failed.
```

Example 4: Editing a List of Fast-Start Failover Targets

The following examples show how to specify a list of fast-start failover targets.

```
DGMGRL> EDIT DATABASE db1 SET PROPERTY FastStartFailoverTarget='db2,  
db3';
```

```
DGMGRL> EDIT DATABASE db2 SET PROPERTY  
FastStartFailoverTarget='db1,db3';
```

```
DGMGRL> EDIT DATABASE db3 SET PROPERTY FastStartFailoverTarget='db1';
```

EDIT DATABASE (Parameter)

The `EDIT DATABASE (Parameter)` command sets the specified initialization parameter for the named database.

Format

```
EDIT DATABASE database_name SET PARAMETER parameter-name = value  
[initialization parameter options]
```

Command Parameters

database-name

The name of the database for which you want to change a parameter value.

parameter-name

The name of the existing database initialization parameter that must be modified.

value

The new value for the parameter.

initialization parameter options

Additional initialization parameter options must be enclosed within single quotes. Use one or both of the following options:

- **SCOPE**: Set one of the following values for scope: `SPFILE`, `MEMORY`, or `BOTH`. The default value is `BOTH`. If the specified parameter is a static parameter, then set `SCOPE= SPFILE`.
- **SID**: Specify the name of a database instance for which the parameter must be set. If the specified parameter must be set for all instances, set `SID='*'`.

Usage Notes

The database must be available when this command is run.

Command Example

The following example edits the initialization parameter `log_archive_trace` for the database named `North_sales` and sets its value to 1. The `SCOPE` setting specifies that

the parameter must be changed in both the memory and in the database initialization parameter file.

```
DGMGRL> EDIT DATABASE 'North_sales' SET PARAMETER log_archive_trace = 1  
'SCOPE = BOTH';
```

EDIT DATABASE (Rename)

The `EDIT DATABASE (Rename)` command changes the name used by the broker to refer to the specified database.

Format

```
EDIT DATABASE database-name RENAME TO new-database-name;
```

Command Parameters

database-name

The name of the database that you want to change.

new-database-name

The new name of the database.

Usage Notes

- Use this command to track changes to the `DB_UNIQUE_NAME` initialization parameter for this database.

Caution:

The *database-name* must always match the value for that database's `DB_UNIQUE_NAME` initialization parameter.

- This command can only be done when broker management of the database that you are renaming is disabled.

Command Example

The following example shows how to edit and rename a database.

```
DGMGRL> DISABLE DATABASE 'South_Sales_typo';  
Disabled.
```

```
DGMGRL> EDIT DATABASE 'South_Sales_typo' RENAME TO 'South_Sales';  
Succeeded.
```

```
DGMGRL> ENABLE DATABASE 'South_Sales';  
Enabled.
```

EDIT DATABASE (State)

The `EDIT DATABASE (State)` command changes the state of the specified database.

Format

```
EDIT DATABASE database-name SET STATE=state [WITH APPLY  
INSTANCE=instance-name];
```

Command Parameters

database-name

The name of the database for which you want to change the state.

state

The state in which you want the database to be running. The possible states are:

```
TRANSPORT-ON (primary database only)  
TRANSPORT-OFF (primary database only)  
APPLY-ON (physical or logical standby database only)  
APPLY-OFF (physical or logical standby database only)
```

instance-name

The name of the instance you want to become the apply instance if this is an Oracle RAC standby database.

Usage Notes

- If the target state is `APPLY-ON` and this database is currently a physical or logical standby database, the optional `WITH APPLY INSTANCE` clause specifies which instance will become the apply instance.
- If the target state is not `APPLY-ON` or if the database is currently in the primary role, the `WITH APPLY INSTANCE` clause is ignored even if it is specified.
- You cannot change the state of a snapshot standby database.
- All instances of an Oracle RAC database are affected by this database state change.

Command Example

The following examples show how to change the state of a database.

```
DGMGR> EDIT DATABASE 'South_Sales' SET STATE='APPLY-ON';  
Succeeded.
```

EDIT DATABASE RESET (Property)

The `EDIT DATABASE RESET (Property)` command reset the specified property for the named database back to its default value.

Format

```
EDIT DATABASE database-name RESET PROPERTY property-name;
```

Command Parameters

database-name

The name of the database for which you want to reset the property value back to its default.

property-name

The name of an existing database-specific configurable property.

Command Example

The following example shows how to reset the `NetTimeout` property for the database named `South_Sales`.

```
DGMGRL> EDIT DATABASE 'South_Sales' RESET PROPERTY NetTimeout;  
Succeeded.
```

EDIT DATABASE RESET (Parameter)

The `EDIT DATABASE RESET (Parameter)` command resets the specified database initialization parameter for the named database to its default value.

Format

```
EDIT DATABASE database-name RESET PARAMETER parameter-name;
```

Command Parameters

database-name

The name of the database for which you want to reset the specified initialization parameter.

parameter-name

The name of an existing database initialization parameter whose value that must be reset.

Command Example

The following example shows how to reset the `log_archive_trace` parameter for the database named `South_Sales`.

```
DGMGRL> EDIT DATABASE 'South_Sales' RESET PARAMETER log_archive_trace;  
Succeeded.
```

EDIT FAR_SYNC

The `EDIT FAR_SYNC` command changes the name, properties, or initialization parameters of a far sync instance.

Format

```
EDIT FAR_SYNC far_sync_instance_name RENAME TO  
new_far_sync_instance_name;
```

```
EDIT FAR_SYNC far_sync_instance_name SET PROPERTY property_name =  
value;
```

```
EDIT FAR_SYNC far_sync_instance_name SET PARAMETER  
parameter_name=value | initialization parameter options;
```

Command Parameters

far_sync_instance_name

The name of the far sync instance for which you want to edit information. It must match (case-insensitive) the value of the corresponding database `DB_UNIQUE_NAME` initialization parameter.

new_far_sync_instance_name

The new name of the far sync instance.

property_name

The name of an existing far sync instance-specific configurable property.

parameter_name

The name of the existing database initialization parameter that must be modified.

value

The new value for the property or parameter.

initialization parameter options

Additional options include the following:

- **SCOPE**: Set one of the following values for scope: `SPFILE`, `MEMORY`, or `BOTH`. The default value is `BOTH`. If the specified parameter is a static parameter, then set `SCOPE=SPFILE`.
- **SID**: Specify the name of a database instance for which the parameter must be set. If the specified parameter must be set for all instances, set `SID='*'`.

Command Examples

The following example renames a far sync instance named `chicago` to the name `dallas`.

```
DGMGRL> DISABLE FAR_SYNC 'chicago_typo'; EDIT FAR_SYNC 'chicago_typo' RENAME TO  
'chicago'; ENABLE FAR_SYNC 'chicago';
```

The following example resets the initialization parameter of a far sync instance named `chicago`.

```
DGMGRL> DISABLE FAR_SYNC 'chicago'; EDIT FAR_SYNC 'chicago' SET  
log_archive_trace=1; ENABLE FAR_SYNC 'chicago';
```

EDIT FAR_SYNC RESET (Property)

The `EDIT FAR_SYNC RESET (Property)` command resets the specified property for the named far sync instance to its default value.

Format

```
EDIT FAR_SYNC far_sync_instance_name RESET PROPERTY property-name;
```

Command Parameters

far_sync_instance_name

The name of the far sync instance for which you want to edit information. It must match (case-insensitive) the value of the corresponding database `DB_UNIQUE_NAME` initialization parameter.

property_name

The name of the property to be reset to its default value.

Command Example

The following example shows how to reset the `ReopenSecs` property back to its default value for the far sync instance named `dallas`.

```
DGMGRL> EDIT FAR_SYNC 'dallas' RESET PROPERTY ReopenSecs;
```

EDIT FAR_SYNC RESET (Parameter)

The `EDIT FAR_SYNC RESET PARAMETER (Parameter)` command resets the specified database initialization parameter for the named far sync instance to its default value.

Format

```
EDIT FAR_SYNC far_sync_instance_name RESET PARAMETER parameter-name;
```

Command Parameters

far_sync_instance_name

The name of the far sync instance for which you want to edit information. It must match (case-insensitive) the value of the corresponding database `DB_UNIQUE_NAME` initialization parameter.

parameter_name

The name of the database initialization parameter to be reset to its default value.

Command Example

The following example shows how to reset the `log_filename_convert` initialization parameter to its default value for the far sync instance named `dallas`.

```
DGMGRL> EDIT FAR_SYNC 'dallas' RESET PARAMETER log_filename_convert;
```

EDIT PLUGGABLE DATABASE (State)

The `EDIT PLUGGABLE DATABASE` command changes the state of the specified pluggable database (PDB).

Format

```
EDIT PLUGGABLE DATABASE pdb-name AT database-name SET STATE = state-name;
```

Parameters

Parameter	Description
<code>pdb-name</code>	Name of the PDB for which you want to change the state.
<code>database-name</code>	Name of the database that contains the PDB specified by <code>pdb-name</code> .
<code>state-name</code>	State to which the PDB must be transitioned. Permitted states are the following: <ul style="list-style-type: none"> <code>APPLY-ON</code>: Starts redo apply from the specified PDB. <code>APPLY-OFF</code>: Stops redo apply from the specified PDB.

Usage Notes

- The specified PDB must exist in the database specified by `database-name` and must have been configured using the `ADD PLUGGABLE DATABASE` command.

Examples

Example 10-7 Stopping Redo Apply at a Target PDB

This example stops redo apply for a target PDB named `dgpdb_sales` that is contained in the target database `newyork`.

```
DGMGRL> EDIT PLUGGABLE DATABASE dgpdb_sales AT newyork SET STATE=APPLY-OFF;
Succeeded.
```

EDIT RECOVERY_APPLIANCE (Property)

The `EDIT RECOVERY_APPLIANCE (Property)` command changes the value of the property for the named Zero Data Loss Recovery Appliance (Recovery Appliance).

Format

```
EDIT object name SET PROPERTY property-name = value;
```

Command Parameters

object name

The name of the Recovery Appliance for which you want to change a property value.

property-name

The name of an existing Recovery Appliance-specific property. Valid properties are as follows:

- DGConnectIdentifier
- LogXptMode
- DelayMins
- Binding
- MaxFailure
- ReopenSecs
- NetTimeout
- RedoCompression
- LogShipping
- InconsistentProperties
- InconsistentLogXptProps
- AlternateLocation

value

The new value for the property.

Command Example

The following example shows an example of editing a configurable property.

```
DGMGRL> EDIT RECOVERY_APPLIANCE 'EnterpriseRecoveryAppliance' SET PROPERTY  
'ReopenSecs'=300;  
Property "ReopenSecs" updated
```

EDIT RECOVERY_APPLIANCE (Parameter)

The `EDIT RECOVERY_APPLIANCE (Parameter)` command sets the specified initialization parameter for the named Zero Data Loss Recovery Appliance (Recovery Appliance).

Format

```
EDIT RECOVERY_APPLIANCE object_name SET PARAMETER parameter-name = value |  
initialization parameter options;
```

Command Parameters

object-name

The name of the Recovery Appliance for which you want to change a parameter value.

parameter-name

The name of the existing database initialization parameter that must be modified.

value

The new value for the initialization parameter.

initialization parameter options

Additional options include the following:

- **SCOPE**: Set one of the following values for scope: SPFILE, MEMORY, or BOTH. The default value is BOTH. If the specified parameter is a static parameter, then set `SCOPE=SPFILE`.
- **SID**: Specify the name of a database instance for which the parameter must be set. If the specified parameter must be set for all instances, set `SID='*'`.

Command Example

The following example edits the initialization parameter for the Recovery Appliance EnterpriseRecoveryAppliance.

```
DGMGRL> EDIT RECOVERY_APPLIANCE EnterpriseRecoveryAppliance' SET  
PARAMETER log_archive_trace = 1 SCOPE='spfile';
```

EDIT RECOVERY_APPLIANCE (Rename)

The `EDIT RECOVERY_APPLIANCE (Rename)` command changes the name used by the broker to refer to the specified Recovery Appliance, as recorded in that Recovery Appliance's profile in the broker configuration.

Format

```
EDIT RECOVERY_APPLIANCE object name RENAME TO new object name;
```

Command Parameters**object name**

The name of the Recovery Appliance that you want to change.

new object name

The new name of the Recovery Appliance.

Usage Notes

- Use this command to track changes to the `DB_UNIQUE_NAME` initialization parameter for this Recovery Appliance.

Caution:

The name of the Recovery Appliance must always match the value of the `DB_UNIQUE_NAME` initialization parameter for that Recovery Appliance.

Command Example

The following example shows how to edit and rename a Recovery Appliance.

```
DGMGRL> EDIT RECOVERY_APPLIANCE 'EnterpriseRecoveryAppliance_typo'  
RENAME TO 'EnterpriseRecoveryAppliance';  
Succeeded.
```

EDIT RECOVERY_APPLIANCE RESET (Property)

The `EDIT RECOVERY_APPLIANCE RESET (Property)` command resets the specified property for the named Recovery Appliance to its default value.

Format

```
EDIT RECOVERY_APPLIANCE object name RESET PROPERTY property-name ;
```

Command Parameters

object name

The name of the Recovery Appliance for which you want to reset the property value back to its default.

property-name

The name of an existing database-specific configurable property.

Command Example

The following example shows how to reset the `ReopenSecs` property back to its default value for the Recovery Appliance named `South_Sales`.

```
DGMGRL> EDIT DATABASE 'South_Sales' RESET PROPERTY ReopenSecs;  
Succeeded.
```

EDIT RECOVERY_APPLIANCE RESET (Parameter)

The `EDIT RECOVERY_APPLIANCE RESET (Parameter)` command resets the specified database initialization parameter for the named Recovery Appliance to its default value.

Format

```
EDIT RECOVERY_APPLIANCE object name RESET PARAMETER parameter-name ;
```

Command Parameters

object name

The name of the Recovery Appliance for which you want to reset the initialization parameter value to its default value.

parameter-name

The name of the database initialization parameter that must be reset.

Command Example

The following example shows how to reset the `db_filename_convert` initialization parameter to its default value for the Recovery Appliance named `EnterpriseRecoveryAppliance`.

```
DGMGRL> EDIT RECOVERY APPLIANCE 'EnterpriseRecoveryAppliance' RESET PARAMETER
db_filename_convert;
Succeeded.
```

ENABLE CONFIGURATION

The `ENABLE CONFIGURATION` command enables the broker to manage the broker configuration, including all of its databases.

Format

```
ENABLE CONFIGURATION [ALL];
```

Command Parameters

None.

Usage Notes

- Use the `SHOW CONFIGURATION` command to enable broker management of the primary database and all members of the configuration, if these members are not explicitly disabled by the user.
- To issue this command, you must connect to a database whose control file role is primary.
- By default, broker management of the configuration's databases is enabled in the `TRANSPORT-ON` state with redo transport services turned on at the primary database and `APPLY-ON` with log apply services started at the standby databases. Far sync instances will be enabled such that they receive redo data and send redo data. You can change the state of a database using the [EDIT DATABASE \(State\)](#) command, but not when the database or the entire configuration is disabled. You cannot change the state of a far sync instance.
- Use this command to update the roles stored in the broker configuration if a failover or switchover was performed using SQL*Plus instead of DGMGRL or Cloud Control.
- Use the [SHOW CONFIGURATION](#) command to display information about the configuration.
- Include the `ALL` keyword to enable all configurations. Use this to start redo transport between the source database to the target database after you create a DB PDB configuration.

Command Example

The following example enables management of a broker configuration.

```
DGMGRL> ENABLE CONFIGURATION;  
Enabled.
```

The following command enables management of all configured broker configurations that contain one or more PDBs. It begins redo transport from the source database to the target database.

```
DGMGRL> ENABLE CONFIGURATION ALL;  
Enabled "MyConfig1" with primary database "boston".  
Enabled "MyConfig2" with primary database "newyork".
```

ENABLE DATABASE

The `ENABLE DATABASE` command enables broker management of the specified standby database.

Caution:

Do not issue the `ENABLE DATABASE` command on a standby database that needs to be reinstated. See [Reenabling Disabled Databases After a Role Change](#) for more details.

Format

```
ENABLE DATABASE database-name;
```

Command Parameters

database-name

The name of the standby database for which you want to enable broker management.

Usage Notes

- You must connect to the primary database or to an already enabled standby database to issue this command.
- A standby database may have been disabled by the broker as a consequence of a prior failover or switchover operation. See [Reenabling Disabled Databases After a Role Change](#) to understand how the database can be reinstated or re-created.
- By default, broker management of the physical or logical standby database is enabled in the `APPLY-ON` state with log apply services enabled. You can change the state of the standby database using the [EDIT DATABASE \(State\)](#) command, but only when the database is enabled.
- Use the [SHOW DATABASE](#) command to display information about the database.
- For an Oracle RAC database, only one instance is required to be started and mounted for this command to succeed.

Command Example

The following example shows how to enable a database named `South_Sales`.

```
DGMGR> ENABLE DATABASE 'South_Sales';  
Enabled.
```

ENABLE FAR_SYNC

The `ENABLE FAR_SYNC` command enables broker management of the specified far sync instance.

Format

```
ENABLE FAR_SYNC far_sync_instance_name;
```

Command Parameters

far_sync_instance_name

The name of the far sync instance for which you want to enable broker management.

Command Example

The following example enables broker management of a far sync instance named `dallas`.

```
DGMGR> ENABLE FAR_SYNC 'dallas';
```

ENABLE FAST_START FAILOVER

The `ENABLE FAST_START FAILOVER` command enables the broker to fail over to a specifically-chosen standby database in the event of loss of the primary database, without requiring any manual steps.

See [Enabling Fast-Start Failover](#) for complete information.

Format

```
ENABLE FAST_START FAILOVER [OBSERVE ONLY];
```

Command Parameters

OBSERVE ONLY: All observers started before or after this command is issued will run in observe-only mode.

Usage Notes

- The prerequisites described in [Prerequisites for Enabling Fast-Start Failover](#) must be met before you issue this command to enable fast-start failover.
- Issuing the `ENABLE FAST_START FAILOVER` command does not trigger a failover, it only allows the observer that is monitoring the configuration to initiate a fast-start failover if conditions warrant a failover.
- You can enable fast-start failover while connected to any database in the broker configuration.

- If you do not start the observer after you have enabled fast-start failover, the ORA-16819 warning is displayed for the primary and target standby databases. For example:

```
DGMGRL> SHOW DATABASE 'South_Sales';
Database - South_Sales

Role:                PRIMARY
Intended State:      TRANSPORT-ON
Instance(s):
  south_sales1

Database Warning(s):
  ORA-16819: fast-start failover observer not started

Database Status:
WARNING
```

- To enable fast-start failover for a broker configuration with multiple standby databases, the `FastStartFailoverTarget` configuration property on the primary database must specify one or more viable target standby databases. Both the primary database and the target standby databases must have:
 - Standby redo logs configured
 - Redo transport must be properly configured at both databases for the configured protection mode

Oracle also recommends Flashback Database be enabled on both the primary and standby databases to allow for reinstatement of the old primary database after a failover. If it is not enabled, then you will receive a warning when you enable fast-start failover:

```
DGMGRL> ENABLE FAST_START FAILOVER;
Warning: ORA-16827: Flashback Database is disabled
```

Task 2 in [Enabling Fast-Start Failover](#), and `FastStartFailoverTarget` provide more information about the `FastStartFailoverTarget` configuration property.

- Once you have enabled fast-start failover, you must comply with the restrictions described in [Restrictions When Fast-Start Failover is Enabled](#).

Command Examples

Example 1: Enabling a Fast-Start Failover

The following example enables fast-start failover.

```
DGMGRL> ENABLE FAST_START FAILOVER;
Enabled in Zero Data Loss Mode.
```

Example 2: Successful Enabling of Fast-Start Failover

The following example shows that fast-start failover was successfully enabled when the configuration is operating in maximum performance mode.

```
DGMGRL> SHOW FAST_START FAILOVER;

Fast-Start Failover: Enabled in Zero Data Loss Mode

Protection Mode: MaxAvailability
```

```

Lag Limit: 0 seconds

Threshold:                180 seconds
Ping Interval: 3000 milliseconds
Ping Retry: 0
Active Target:            South_Sales
Potential Targets:       "South_Sales"
                          South_Sales valid
Observer:                 (none)
Shutdown Primary:        TRUE
Auto-reinstate:          TRUE
Observer Reconnect:      (none)
Observer Override:       FALSE

```

Configurable Failover Conditions

```

Health Conditions:
  Corrupted Controlfile      YES
  Corrupted Dictionary       YES
  Inaccessible Logfile       NO
  Stuck Archiver             NO
  Datafile Write Errors      YES

Oracle Error Conditions:
  (none)

```

Related Topics

- [Scenario 16: Using the Observe-only Mode for Fast-Start Failover](#)
The observe-only mode enables you to test the impact of using fast-start failover in your configuration, without making any actual changes to the configuration. You can use the DGMGRL commands or data dictionary views to verify the observe-only mode setting.

ENABLE FAST_START FAILOVER CONDITION

The `ENABLE FAST_START FAILOVER CONDITION` command specifies additional conditions for which a fast-start failover should be performed.

Format

```
ENABLE FAST_START FAILOVER CONDITION value;
```

Command Parameters

value

Possible values are those described in the `SHOW FAST_START FAILOVER` command as health conditions. The Oracle error `ORA-00240` can also be named as a condition by specifying `240` as the value.

Usage Notes

- [Table 10-2](#) lists some examples of health conditions maintained by the database health-check facility.

- An error is raised if the specified value is not recognized or if the condition has already been set.
- **Table 10-2 Examples of Health Conditions**

Health Condition	Description
Datafile Write Errors	If fast-start failover is enabled and the Datafile Write Errors condition is specified, then a fast-start failover is initiated if write errors are encountered in any data files, including temp files, system data files, and undo files.
"Corrupted Controlfile"	Corrupted controlfile. This condition is enabled by default.
"Corrupted Dictionary"	Dictionary corruption of a critical database object. This condition is enabled by default.
"Inaccessible Logfile"	LGWR is unable to write to any member of a log group due to an I/O error.
"Stuck Archiver"	Archiver is unable to archive a redo log because device is full or unavailable.

- You can display these configurable conditions with the `SHOW FAST_START FAILOVER` command.

Command Examples

Example 1

The following example specifies that a fast-start failover should be done if a corrupted controlfile is detected.

```
ENABLE FAST_START FAILOVER CONDITION "Corrupted Controlfile";
```

Example 2

The following example specifies that a fast-start failover should be done if an ORA-00240 error is raised.

```
ENABLE FAST_START FAILOVER CONDITION 240;
```

Example 3

The following example displays output that shows the condition Datafile Write Errors.

```
DGMGRL> SHOW FAST_START FAILOVER;
```

```
Fast-Start Failover: DISABLED

Threshold:          180 seconds
Ping Interval:     3000 milliseconds
Ping Retry:        0
Active Target:     (none)
Potential Targets: "South_Sales"
  South_Sales      valid
Observer:          (none)
Lag Limit:         300 seconds
Shutdown Primary: TRUE
Auto-reinstate:   TRUE
Observer Reconnect: (none)
Observer Override: FALSE
```

Configurable Failover Conditions

```
Health Conditions:
Corrupted Controlfile      YES
Corrupted Dictionary      YES
Inaccessible Logfile      NO
Stuck Archiver            NO
Datafile Write Errors     YES
```

```
Oracle Error Conditions:
(none)
```

ENABLE RECOVERY_APPLIANCE

The `ENABLE RECOVERY_APPLIANCE` command enables broker management of the specified Zero Data Loss Recovery Appliance (Recovery Appliance).

Format

```
ENABLE RECOVERY_APPLIANCE object name;
```

Command Parameters

object name

The name of the Recovery Appliance for which you want to enable broker management.

Usage Notes

- You must connect to the primary database or to an already enabled standby database to issue this command.
- Use the `SHOW RECOVERY_APPLIANCE` command to display information about the Recovery Appliance.

Command Example

The following example shows how to enable a Recovery Appliance named `EnterpriseRecoveryAppliance`.

```
DGMGRL> ENABLE RECOVERY_APPLIANCE 'EnterpriseRecoveryAppliance';Enabled.
```

EXIT

The `EXIT` command exits (quits) the broker's command-line interface.

Format

```
EXIT;
```

Command Parameters

None.

Usage Notes

- This command has the same effect as the `QUIT` command.
- A database connection is not required to execute this command. However, if you are connected, this command breaks the connection.

Command Example

The following example demonstrates how to exit (quit) the command-line interface.

```
DGMGRL> EXIT;
```

EXPORT CONFIGURATION

The `EXPORT CONFIGURATION` command enables you to save the metadata contained in the broker configuration file to a text file. Use this command to maintain an up-to-date copy of the broker configuration metadata.

Format

```
EXPORT CONFIGURATION [TO file-name];
```

Command Parameters

file-name

The name of the file in which the Data Guard broker configuration is saved.

If you omit the `TO file-name` clause, the broker stores the exported configuration using a default file name. The convention used to name the file is `SID_dmon_processID-of-DMON_brkmeta_serial-number.trc`.

For example, if the SID is `orcl` and the process ID of the PMON process is 1234, and the `TO file-name` clause is omitted, the file created when the broker configuration is first exported is named `orcl_dmon_1234_brkmeta_1.trc`. When the broker configuration is next exported, it is stored in a file named `orcl_dmon_1234_brkmeta_2.trc`.

Usage Notes

- The broker stores the exported configuration in the `trace` directory. You cannot specify the directory in which the configuration must be stored.
- When you need to downgrade to an earlier version of the database software, you can export the broker configuration to a file before you downgrade the database software. Subsequently, instead of manually recreating the configuration from scratch, you can import this exported file to recreate the broker configuration after the downgrade is complete.

Command Example

The following example exports the metadata in the broker configuration file to a file named `dg_config.txt` in the `trace` directory.

```
EXPORT CONFIGURATION TO 'dg_config.txt';
```

FAILOVER

The `FAILOVER` command invokes a failover that transitions the named (target) standby database into the role of a primary database.

This type of failover is referred to as a *manual failover*. See [Manual Failover](#) for more information.

 **Note:**

Because a failover results in a transition of a standby database to the primary role, it should be performed when the primary database has failed or is unreachable and cannot be recovered in a timely manner. Failover may or may not result in data loss depending on the protection mode in effect at the time of the failover and whether the target standby database was synchronized with the primary database.

Use the `SWITCHOVER` command if the primary database has not failed and you want the current primary database and a standby database to switch roles with no data loss.

Format

```
FAILOVER TO database-name [IMMEDIATE];
```

Command Parameters

database-name

The name of a physical, logical, or snapshot standby database that you want to fail over to the primary database role.

Usage Notes

- Always try to perform a complete failover first unless Redo Apply has stopped at the failover target due to an `ORA-752` or `ORA-600 [3020]` error. If one of these errors has occurred, then before proceeding follow the guidelines in "Resolving `ORA-752` or `ORA-600 [3020]` During Standby Recovery" in My Oracle Support Note 1265884.1 at <http://support.oracle.com>. An immediate failover should only be performed when a complete failover is unsuccessful or in the error case just noted.
- The specified standby database must be enabled before the primary database fails. However, an enabled standby database that was shut down can be a candidate for the failover operation. In this case, restart the standby database using `DGMGRL STARTUP` command, then issue the `FAILOVER` command.
- The failover operates on the specified standby database and changes its role to a primary database. Bystander standby databases (those not involved in the failover) remain in the standby role.
- Before you issue the `FAILOVER` command, verify that you are connected to the standby database that will become the new primary database. If necessary, issue a `CONNECT` command to connect to the standby database to which you want to failover.
- If the `FAILOVER` command is issued without any options, the standby database chosen as the failover target applies all unapplied redo it has received before changing to the primary role. This is referred to as a complete failover.
- If the broker configuration is operating in maximum protection mode, a manual failover operation will force the protection mode to be maximum performance. The

redo transport service settings are unaffected. You need to restore the desired protection mode for the resulting configuration after the failover operation.

 **Note:**

With fast-start failover, the broker preserves the protection mode that was in effect prior to the failover.

- If the `FAILOVER` command is issued with the `IMMEDIATE` option, no attempt is made to apply any unapplied redo that has been received. This option more likely results in lost application data even when standby redo log files are configured on the standby database. Additionally, any remaining standby databases in the configuration cannot function as such until they are reinstated or re-created. See [Reenabling Disabled Databases After a Role Change](#) for more information.
- You can perform a manual failover or set up the broker to perform a fast-start failover. See the `ENABLE FAST_START FAILOVER` command for information about allowing the broker to automatically invoke failover, when conditions warrant a failover.
- If fast-start failover is enabled, you can perform a *complete* manual failover only to the fast-start failover target standby database and only if the fast-start failover target standby database is synchronized with, or within the lag limit of, the primary database, and only when the observer is started. You cannot perform an *immediate* manual failover when fast-start failover is enabled.
- If Flashback Database was enabled on the former (failed) primary database prior to the failover, the former primary database can be reinstated using the broker's `REINSTATE` command (see the `REINSTATE DATABASE` command).

If failover was performed to a physical standby database, any other physical standby databases that were disabled by the failover can be reinstated if Flashback Database was enabled on the standby database and there are sufficient flashback logs available. See [Reenabling Disabled Databases After a Role Change](#) for step-by-step instructions.

- The original primary database can only participate in the configuration as a standby database after it is reinstated or re-created.

 **Caution:**

You should shut down the original primary database if it still has any active instances running prior to failing over.

 **See Also:**

[Reenabling Disabled Databases After a Role Change](#) about reenabling the original primary database so that it could serve as a standby database to the primary database

Command Example

The following example performs a failover in which the standby database, `South_Sales`, transitions to the primary role:

```
DGMGRL> FAILOVER TO 'South_Sales';
Performing failover NOW, please wait...
Failover succeeded, new primary is "South_Sales"

DGMGRL> SHOW CONFIGURATION;
Configuration - DRSolution

Protection Mode: MaxPerformance
Members:
  South_Sales - Primary database
  North_Sales - Physical standby database (disabled)
  ORA-16661: the standby database needs to be reinstated

Fast-Start Failover: DISABLED

Configuration Status:
WARNING
```

FAILOVER TO PLUGGABLE DATABASE

The `FAILOVER TO PLUGGABLE DATABASE` command invokes a failover that transitions the named target PDB into the role of a source PDB.

Syntax

```
FAILOVER TO PLUGGABLE DATABASE pdb-name AT database-name [IMMEDIATE];
```

Parameters

Parameter	Description
<code>pdb-name</code>	Name of the target PDB to that you want to fail over to the source PDB role. This must be the same PDB that was used when setting up the source PDB with the <code>ADD PLUGGABLE DATABASE</code> command.
<code>database-name</code>	Name of the target database that contains the target PDB.

Usage Notes

- When there are problems with the source PDB, but the source database is available, perform a manual failover to a target PDB. The target PDB applies all the unapplied redo that it has received and then changes over to a source PDB role. This is referred to as a complete PDB failover.
- When the source database is not accessible, perform immediate failover by using the `FAILOVER PLUGGABLE DATABASE` command with the `IMMEDIATE` option. No attempt is made to apply any unapplied redo that has been received. This option more likely results in lost application data.

- The original source PDB must be manually re-enabled so that it can serve as a target PDB for the new source PDB.

Example 10-8 Performing Failover of a PDB

The following example performs a PDB failover in which the target PDB `sales`, contained in the target database `newyork`, transitions to the source PDB.

```
DGMGRL> FAILOVER TO PLUGGABLE DATABASE sales AT newyork;
Verifying conditions for Failover...
  Source pluggable database is 'SALESB' at database 'boston'
Performing FAILOVER now, please wait...
  Converting 'SALESB' to standby role...
  Waiting for 'SALES' to recover all redo data...
  Stopping recovery at 'SALES'...
  Converting 'SALES' to primary role...
  Opening new primary 'SALES'...
Failover succeeded, new primary is "sales".
```

HELP

The `DISPLAY` command displays online help for the Data Guard command-line interface.

Format

```
HELP [command_name];
```

Command Parameters

command_name

The command for which you want to display help information. If you do not specify a command, then all commands are listed. The following commands are available:

@	Execute DGMGRL script file
!	Host operating system command
/	Repeat the last command
--	Comment to be ignored by DGMGRL
add	Adds a member to the broker configuration
connect	Connects to an Oracle database instance
convert	Converts a database from one type to another
create	Creates a broker configuration
disable	Disables a configuration, a member, or fast-start failover
edit	Edits a configuration or a member
enable	Enables a configuration, a member, or fast-start failover
exit	Exits the program
failover	Changes a standby database to be the primary database
help	Displays description and syntax for a command
host	Host operating system command
migrate	Migrate a pluggable database from one configuration to another.
quit	Exits the program
reinstat	Changes a database marked for reinstatement into a viable
standby	

rem	Comment to be ignored by DGMGRL
remove	Removes a configuration or a member
set	Set a property to a specified value
show	Displays information about a configuration or a member
shutdown	Shuts down a currently running Oracle database instance
spool	store input and output of DGMGRL CLI in a file
sql	Executes a SQL statement
start	Starts the fast-start failover observer
startup	Starts an Oracle database instance
stop	Stops the fast-start failover observer
switchover	Switches roles between a primary and standby database
validate	Performs an exhaustive set of validations for a database

Enter `help command_name` to see syntax for individual commands.

Usage Notes

- A database connection is not required to execute this command.

Command Example

The following example gets help on the `EDIT` commands.

```
DGMGRL> HELP EDIT
```

HOST or ! (exclamation point)

The DGMGRL `HOST` and `!` commands allow you to execute operating system command(s) directly through the DGMGRL console without leaving DGMGRL.

The `HOST` command and the `!` command have the same functionality. They allow you to submit operating system commands while you are logged in to DGMGRL. The DGMGRL prompt becomes a shell prompt which accepts operating system commands. You can also directly submit individual operating system commands to DGMGRL through the `Host` or `!` command.

Format

```
HOST [command]
```

Or alternatively,

```
!command
```

Command Parameters

command

Represents an operating system command

Usage Notes

- If you simply enter `HOST` without specifying a command, then the DGMGRL console becomes an operating system shell prompt until you issue the `EXIT` command to return to the DGMGRL console.

- The `HOST` and `!` commands take all the content entered on the command line after them as input for the operating system shell prompt. See Command Example 2 below.

Command Examples

Example 1

The following example shows the `HOST` and `!` commands being used to execute an individual operating system command in the `DGMGRL` console.

```
DGMGRL> HOST DATE
Executing operating system command(s):" date"
Fri Oct 23 14:08:42 EDT 2015
DGMGRL>
```

```
DGMGRL> ! DATE
Executing operating system command(s):" date"
Fri Oct 23 14:09:20 EDT 2015
DGMGRL>
```

Example 2

In the following example, both of the `DATE` commands are executed in the operating system shell before control is returned to `DGMGRL`.

```
DGMGRL> ! DATE;DATE;
Executing operating system command(s):" date;date;"
Fri Oct 23 14:11:40 EDT 2015
Fri Oct 23 14:11:40 EDT 2015
DGMGRL>
```

IMPORT CONFIGURATION

The `IMPORT CONFIGURATION` command enables you to import the broker configuration metadata that was previously exported using the `EXPORT CONFIGURATION` command.

Format

```
IMPORT CONFIGURATION FROM file-name;
```

Command Parameters

file-name

The name of file that contains the exported broker configuration metadata.

Usage Notes

- The imported metadata is stored in the in-memory metadata and to either of the broker metadata files specified by `DG_BROKER_CONFIG_FILE1` or `DG_BROKER_CONFIG_FILE2`.
- The specified file name must exist in the `trace` directory.

Command Example

The following command imports configuration metadata stored in the file named `dg_config.txt` in the `trace` directory into the memory and to the broker metadata file.

```
DGMGRL> IMPORT CONFIGURATION FROM 'dg_config.txt';
```

MIGRATE PLUGGABLE DATABASE

The `MIGRATE PLUGGABLE DATABASE` command lets you migrate a pluggable database (PDB) from one multitenant container database (CDB) to another on the same host. You can migrate a PDB from a primary CDB to another primary CDB or failover a PDB from a standby CDB to a primary CDB.

Format

```
MIGRATE PLUGGABLE DATABASE [VERIFY][IMMEDIATE] pdb-name
TO CONTAINER dest-cdb-name
USING XML-description-file
[CONNECT AS { /@dest-cdb-connect-identifer | dest-cdb-user/dest-
cdb-pass@dest-cdb-connect-identifer } ]
[ SECRET secret KEYSTORE IDENTIFIED BY ( EXTERNAL STORE | wallet-
password )];
```

Prerequisites

The `MIGRATE PLUGGABLE DATABASE` command has the following prerequisites:

- The destination CDB must be created and started in such a way that it can access the PDB data files at the same file path as the source CDB.
- The source and destination CDBs must each be in a different Data Guard broker configuration.
- The source CDB can either be a primary database or a physical standby database.
- If the source PDB uses Transparent Data Encryption (TDE), the encryption keys associated with the source PDB must also be migrated. Connect to the database root and use the `ADMINISTER KEY MANAGEMENT` command to export the master encryption keys. The clauses used with this command depend on the PDB mode (isolated mode or united mode) and the type of keystore (software-based keystore or password keystore).

The file containing the exported keys must be accessible to the destination CDB either using Secure File Copy (SCP) or Oracle Automatic Storage Management Cluster File System (Oracle ACFS).

- If the source CDB is a physical standby database:
 - source CDB and destination CDB must be running the same Oracle version and patches.
 - source CDB and destination CDB must have the same setting for the `COMPATIBLE` initialization parameter.
 - PDB to be migrated must be closed on its primary CDB.

- execute the failover
 - if a PDB failover will result in lost data, then you must specifically choose to execute the failover by using the `IMMEDIATE` option.
- If the source CDB is a primary database, then
 - the destination CDB cannot be running a lower version of Oracle.
 - the setting of the `COMPATIBLE` initialization parameter cannot be set to a lower value on the destination CDB than on the source CDB.
- The services for the PDB must be stopped and removed from the Oracle Clusterware repository of the source database to be migrated. This should be done for all databases in the Data Guard broker configuration of the source database.
- The destination CDB must be open.
- You must have `SYSDBA` privilege for both the source and destination CDBs.

Command Parameters

pdb-name

The name of the PDB to be migrated.

dest-cdb-name

The database unique name of the CDB to receive the PDB to be migrated.

XML-description-file

An XML file that contains the description of the PDB to be migrated. This file is automatically created by the SQL statements executed by the `MIGRATE PLUGGABLE DATABASE` command.

dest-cdb-user

The user name of the user that has `SYSDBA` access to the destination CDB.

dest-cdb-pass

The password associated with the user name specified for *dest-cdb-user*.

dest-cdb-connect-identifier

An Oracle Net connect identifier used to reach the destination CDB.

secret

The password that was used to encrypt the export file containing the exported encryption keys of the source PDB. This password would have been specified as part of the `ADMINISTER KEY MANAGEMENT` command, when exporting the PDB encryption keys.

wallet-password

The password of the keystore containing the encryption keys. This is required if the source PDB was encrypted using a password keystore.

Usage Notes

- By default, when this command is used for PDB failover, the failover attempt is rejected if there is a possibility of data loss. You can override this default behavior by using the `IMMEDIATE` option.
- The `IMMEDIATE` option is ignored if the source database is a primary database.
- The `VERIFY` option performs a pre-check to determine if the PDB can be migrated successfully.

- If a connect identifier is specified, then database credentials are used to authenticate the user on the destination CDB.
- Operating system credentials cannot be used to authenticate the user on the destination CDB. A connect identifier must be specified; a slash (/) is not supported.
- For cases in which a slash (/) is used to specify a connect identifier (for example, /@boston), the credentials are fetched from the wallet.
- The following options are available if you want to specify a connect string:
 - /@*dest-cdb-connect-identifier* (credentials are fetched from the wallet)
 - *dest-cdb-user/dest-cdb-pass@dest-cdb-connect-identifier* (uses database credentials)
- To prevent the password from being visible on the command line, specify only a user name with a connect identifier. You will then be prompted for a password. The following options are supported when you supply only a user name and connect identifier:
 - *dest-cdb-user@dest-cdb-connect-identifier* (uses database credentials)
 - *dest-cdb-user/@dest-cdb-connect-identifier* (uses database credentials)
- If you omit the connect string entirely from the command line, then you will be prompted for a user name and password. The following options are supported:
 - /@*dest-cdb-connect-identifier* (no prompt for password, credentials are fetched from the wallet)
 - *dest-cdb-user@dest-cdb-connect-identifier* (uses database credentials)
 - *dest-cdb-user/@dest-cdb-connect-identifier* (uses database credentials)

Command Examples

Example 1: Migrating a PDB From a Primary CDB

```
DGMGRL> MIGRATE PLUGGABLE DATABASE REGION1 TO CONTAINER
NORTH_SALES_NEW USING REGION1.xml
CONNECT AS sys@NORTH_SALES_NEW;
Connected to "NORTH_SALES_NEW"
Connected.
```

```
Beginning migration of pluggable database REGION1.
Source multitenant container database is NORTH_SALES.
Destination multitenant container database is NORTH_SALES_NEW.
```

```
Closing pluggable database REGION1 on all instances of multitenant
container database NORTH_SALES.
Unplugging pluggable database REGION1 from multitenant container
database NORTH_SALES.
Pluggable database description will be written to REGION1.xml.
Dropping pluggable database REGION1 from multitenant container
database NORTH_SALES.
Creating pluggable database REGION1 on multitenant container database
NORTH_SALES_NEW.
Opening pluggable database REGION1 on all instances of multitenant
```

container database NORTH_SALES_NEW.
Succeeded.

Example 2: Failing over a PDB from a Physical Standby

```
DGMGRL> MIGRATE PLUGGABLE DATABASE REGION1 TO CONTAINER SOUTH_SALES_NEW
USING REGION1.xml CONNECT AS sys@SOUTH_SALES_NEW;
Connected to "SOUTH_SALES_NEW"
Connected.
```

Beginning migration of pluggable database REGION1.
Source multitenant container database is SOUTH_SALES.
Destination multitenant container database is SOUTH_SALES_NEW.

Continuing with migration of pluggable database REGION1 to multitenant
container database SOUTH_SALES_NEW.
Stopping Redo Apply services on source multitenant container database
SOUTH_SALES.
Succeeded.
Opening database SOUTH_SALES.
Opening pluggable database REGION1 on source multitenant container database
SOUTH_SALES to prepare for migration.
Pluggable database description will be written to REGION1.xml.
Closing pluggable database REGION1 on all instances of multitenant container
database SOUTH_SALES.
Disabling media recovery for pluggable database REGION1.
Closing database SOUTH_SALES.
Restarting redo apply services on source multitenant container database
SOUTH_SALES.
Succeeded.
Creating pluggable database REGION1 on multitenant container database
SOUTH_SALES_NEW.
Opening pluggable database REGION1 on all instances of multitenant container
database SOUTH_SALES_NEW.
Unplugging pluggable database REGION1 from multitenant container database
NORTH_SALES.
Dropping pluggable database REGION1 from multitenant container database
NORTH_SALES.
Succeeded.

Example 10-9 Performing a Pre-Migration Verification of a Source PDB

```
DGMGRL> MIGRATE PLUGGABLE DATABASE VERIFY IMMEDIATE MYPDB TO CONTAINER
tgt_cdb USING mypdb.xml CONNECT AS sys@tgtcdb_tns ;
Connected.
```

Beginning verification of migration of pluggable database MYPDB.
Source multitenant container database is src_cdb.
Destination multitenant container database is tgt_cdb.

Potential data loss is possible for pluggable database MYPDB.
Opening database in restricted mode on this instance of database src_cdb.
Opening pluggable database MYPDB on this instance of multitenant container
database src_cdb.

```
Pluggable database description will be written to mypdb.xml.  
No plug-in violations found.  
Closing database src_cdb.  
Verification of pluggable database MYPDB complete, no  
incompatibilities found.
```

PREPARE DATABASE FOR DATA GUARD

The `PREPARE DATABASE FOR DATA GUARD` command configures a database for use as a primary database in a Data Guard broker configuration. Database initialization parameters are set to recommended values.

Format

```
PREPARE DATABASE FOR DATA GUARD [WITH DB_UNIQUE_NAME IS db-unique-name] [DB_RECOVERY_FILE_DEST IS directory-location]  
[DB_RECOVERY_FILE_DEST_SIZE IS size] [BROKER_CONFIG_FILE_1 IS broker-config-file-1-location] [BROKER_CONFIG_FILE_2 IS broker-config-file-2-location];
```

Command Parameters

db_unique_name

The value for the `DB_UNIQUE_NAME` initialization parameter. If the initialization parameter has been set to a different value, the existing value is replaced with the value specified by `db_unique_name`. If this parameter is not specified, the `DB_UNIQUE_NAME` parameter is set to the value of the `DBNAME` parameter.

directory-location

The directory name for the `DB_RECOVERY_FILE_DEST` initialization parameter, which represents the fast recovery area location. The specified directory must be accessible by all instances of a RAC database.

This parameter can be omitted if a local archive destination is set. However, if the `DB_RECOVERY_FILE_DEST` initialization parameter has not been set and no local archive destination has been set, specifying this parameter is mandatory.

If `directory_location` is specified, a `log_archive_dest_n` initialization parameter is set to the value `USE_DB_RECOVERY_FILE_DEST`. This is done whether or not there is a local archive destination already set.

size

A size value for the `DB_RECOVERY_FILE_DEST` initialization parameter. This parameter is mandatory if the `DB_RECOVERY_FILE_DEST` is specified.

broker-config-file-1-location

A file location that is used to set the `DG_BROKER_CONFIG_FILE1` initialization parameter. The file location specified must be accessible by all instances of a RAC database. This is an optional command parameter.

broker-config-file-2-location

A file location that is used to set the `DG_BROKER_CONFIG_FILE2` initialization parameter. The file location specified must be accessible by all instances of a RAC database. This is an optional command parameter.

Prerequisites

You must connect to the primary database as a user with the `SYSDG` or `SYSDBA` privilege.

Usage Notes

- Database versions starting from Oracle Database 12c Release 2 are supported.
- For a single-instance database, if a server parameter file does not exist, it is created using the current in-memory parameter settings and stored in the default location.
- This command sets the following initialization parameters, as per the values recommended for the Maximum Availability Architecture (MAA):
 - `DB_FILES=1024`
 - `LOG_BUFFER=256M`
 - `DB_BLOCK_CHECKSUM=TYPICAL`
If this value is already set to `FULL`, the value is left unchanged.
 - `DB_LOST_WRITE_PROTECT=TYPICAL`
If this value is already set to `FULL`, the value is left unchanged.
 - `DB_FLASHBACK_RETENTION_TARGET=120`
If this parameter is already set to a non-default value, it is left unchanged.
 - `PARALLEL_THREADS_PER_CPU=1`
 - `DG_BROKER_START=TRUE`
- This command enables archivelog mode, enables force logging, enables Flashback Database, and sets the RMAN archive log deletion policy to `SHIPPED TO ALL STANDBY`.
- If standby redo logs do not exist in the primary database, they are added. If the logs exist and are misconfigured, they are deleted and recreated.

Command Example

The following example prepares a database with the name `boston` for use as a primary database. The recovery destination is `$ORACLE_BASE_HOME/dbs`.

```
DGMGRL> PREPARE DATABASE FOR DATA GUARD
WITH DB_UNIQUE_NAME IS boston
DB_RECOVERY_FILE_DEST IS "$ORACLE_BASE_HOME/dbs/"
DB_RECOVERY_FILE_DEST_SIZE IS "400G"
DG_BROKER_CONFIG_FILE1 IS "$ORACLE_HOME/dbs/file1.dat"
DG_BROKER_CONFIG_FILE2 IS "$ORACLE_HOME/dbs/file2.dat";
```

```
Preparing database "boston" for Data Guard.
Creating server parameter file (SPFILE) from initialization parameter memory
values.
Database must be restarted after creating the server parameter (SPFILE).
Shutting down database "boston".
Database closed.
Database dismounted.
ORACLE instance shut down.
Starting database "boston" to mounted mode.
ORACLE instance started.
Database mounted.
```

```
Server parameter file (SPFILE) is "ORACLE_BASE_HOME/dbs/spboston.ora".
Initialization parameter DB_UNIQUE_NAME set to 'boston'.
Initialization parameter DB_FILES set to 1024.
Initialization parameter LOG_BUFFER set to 268435456.
Primary database must be restarted after setting static initialization
parameters.
Primary database must be restarted to enable archivelog mode.
Shutting down database "boston".
Database dismounted.
ORACLE instance shut down.
Starting database "boston" to mounted mode.
ORACLE instance started.
Database mounted.
Initialization parameter DB_FLASHBACK_RETENTION_TARGET set to 120.
Initialization parameter DB_BLOCK_CHECKSUM set to 'TYPICAL'.
Initialization parameter DB_LOST_WRITE_PROTECT set to 'TYPICAL'.
Initialization parameter PARALLEL_THREADS_PER_CPU set to 1.
Removing RMAN archivelog deletion policy 1.
Removing RMAN archivelog deletion policy 2.
RMAN configuration archivelog deletion policy set to SHIPPED TO ALL
STANDBY.
Initialization parameter DB_RECOVERY_FILE_DEST_SIZE set to '400G'.
Initialization parameter DB_RECOVERY_FILE_DEST set to
'ORACLE_BASE_HOME/dbs/'.
Initialization parameter DG_BROKER_START set to FALSE.
Initialization parameter DG_BROKER_CONFIG_FILE1 set to
'ORACLE_HOME/dbs/file1.dat'.
Initialization parameter DG_BROKER_CONFIG_FILE2 set to
'ORACLE_HOME/dbs/file2.dat'.
LOG_ARCHIVE_DEST_n initialization parameter already set for local
archival.
Initialization parameter LOG_ARCHIVE_DEST_2 set to
'location=use_db_recovery_file_dest valid_for=(all_logfiles,
all_roles)'.
Initialization parameter LOG_ARCHIVE_DEST_STATE_2 set to 'Enable'.
Initialization parameter STANDBY_FILE_MANAGEMENT set to 'MANUAL'.
Standby log group 4 will be dropped because it was not configured
correctly.
Standby log group 3 will be dropped because it was not configured
correctly.
Adding standby log group size 26214400 and assigning it to thread 1.
Initialization parameter STANDBY_FILE_MANAGEMENT set to 'AUTO'.
Initialization parameter DG_BROKER_START set to TRUE.
Database set to FORCE LOGGING.
Database set to ARCHIVELOG.
Database set to FLASHBACK ON.
Database opened.
```

QUIT

The `QUIT` command quits (exits) the Data Guard command-line interface.

Format

```
QUIT;
```

Command Parameters

None.

Usage Notes

- This command has the same effect as the [EXIT](#) command.
- A database connection is not required to execute this command. However, if you are connected, this command breaks the connection.

Command Example

The following example shows how to quit (exit) the command-line interface.

```
DGMGRL> QUIT;
```

REINSTATE DATABASE

The `REINSTATE DATABASE` command reinstates a database as a new standby database in the broker configuration for the current primary database.

Format

```
REINSTATE DATABASE database-name;
```

Command Parameters

database-name

The name of the database that is to be reinstated in the broker configuration.

Usage Notes

- If the conditions for reinstatement described in [Reinstating the Former Primary Database in the Broker Configuration](#) are not satisfied, the reinstatement will fail with an appropriate error status and the specified database will remain disabled.
- If the *database-name* specified is that of the old primary and fast-start failover is enabled, the old primary database will be reinstated as a standby to the new primary, and the fast-start failover environment will be updated to reflect the availability of the new standby database. It will accept redo data from the new primary database and be the target of a fast-start failover should the new primary database fail. Reinstatement occurs automatically if the observer is running unless the `FastStartFailoverAutoReinstate` configuration property is set to `FALSE`.
- This command does not require that fast-start failover be enabled. It can be used to reinstate an old primary database after a complete manual failover has been performed.

It can also be used to reinstate a bystander standby database that had been disabled after either a complete or immediate failover.

- Issue this command while connected to any database in the broker configuration, except the database that is to be reinstated.

Command Example

The following example reinstates the `North_Sales` database as a standby database in the broker configuration.

```
DGMGRL> REINSTATE DATABASE 'North_Sales';
Reinstating database "North_Sales", please wait...
Reinstatement of database "North_Sales" succeeded
```

REMOVE CONFIGURATION

The `REMOVE CONFIGURATION` command removes the Oracle Data Guard broker configuration and ends broker management of all members in the configuration.

Format

```
REMOVE CONFIGURATION [ PRESERVE DESTINATIONS ];
```

Command Parameters

None.

Usage Notes

- When you remove a broker configuration, management of all of the members associated with that configuration is disabled.
- By default, the command removes the corresponding broker settings of the `LOG_ARCHIVE_DEST_n` initialization parameter on the primary database and the `LOG_ARCHIVE_CONFIG` initialization parameters on all members of the configuration. To preserve these settings, use the `PRESERVE DESTINATIONS` option.
- This command does not remove or affect the actual primary or standby database instances, databases, far sync instances, data files, control files, initialization parameter files, server parameter files, or log files of the underlying Oracle Data Guard configuration.
- You cannot remove the configuration when fast-start failover is enabled.

Command Examples

The following examples show a successful and an unsuccessful `REMOVE CONFIGURATION` command.

Example 1

The following command shows how to remove configuration information from the configuration file.

```
DGMGRL> REMOVE CONFIGURATION;
Removed configuration
DGMGRL> SHOW CONFIGURATION;
Error: ORA-16532: Data Guard broker configuration does not exist
```

Configuration details cannot be determined by DGMGRL

Example 2

The following command is unsuccessful because fast-start failover is enabled.

```
DGMGRL> REMOVE CONFIGURATION;
Error: ORA-16654: fast-start failover is enabled

Failed.

DGMGRL> SHOW CONFIGURATION;

Configuration - DRSolution

  Protection Mode: MaxAvailability
  Members:
    North_Sales - Primary database
    South_Sales - (*) Physical standby database

Fast-Start Failover: Enabled in Zero Data Loss Mode

Configuration status:
SUCCESS
```

REMOVE DATABASE

The `REMOVE DATABASE` command removes the specified standby database from the broker configuration and terminates broker management of that standby database.

Format

```
REMOVE DATABASE database-name [ PRESERVE DESTINATIONS ];
```

Command Parameters

database-name

The name of the standby database that you want to remove from the broker configuration.

Usage Notes

- An error is returned if you specify the name of the primary database in the broker configuration.
- By default, this command removes all references to the specified database from all redo transport initialization parameters at each member of the configuration. To preserve these settings, use the `PRESERVE DESTINATIONS` option.
- This command cannot be executed if fast-start failover is enabled and *database-name* specifies the name of the target standby database.

Command Example

The following example shows how to remove a database from the Oracle Data Guard broker configuration.

```
DGMGRL> SHOW CONFIGURATION;

Configuration - DRSolution
```

```
Protection Mode: MaxPerformance
Members:
  North_Sales - Primary database
  South_Sales - Physical standby database

Fast-Start Failover: DISABLED

Configuration status:
SUCCESS

DGMGRL> REMOVE DATABASE 'South_Sales';
Removed database "South_Sales" from the configuration.
```

Configuration - DRSolution

```
Protection Mode: MaxPerformance
Members:
  North_Sales - Primary database

Fast-Start Failover: DISABLED

Configuration status:
SUCCESS
```

REMOVE FAR_SYNC

The `REMOVE FAR_SYNC` command removes a far sync instance from an Oracle Data Guard broker configuration.

Format

```
REMOVE FAR_SYNC far_sync_instance_name;
```

Command Parameters

far_sync_instance_name

The name of the far sync instance to be removed from the broker configuration. It must match (case-insensitive) the value of the corresponding database `DB_UNIQUE_NAME` initialization parameter.

Usage Notes

- A far sync instance that has its `RedoRoutes` property set cannot be removed.

Command Example

The following example removes a far sync instance named `dallas` from the broker configuration.

```
DGMGRL> REMOVE FAR_SYNC 'dallas';
```

REMOVE INSTANCE

The `REMOVE INSTANCE` command removes the specified instance from the broker configuration.

Format

```
REMOVE INSTANCE instance-name [ON { DATABASE | FAR_SYNC } object-name];
```

Command Parameters

instance-name

The name of the instance (SID) that you want to remove from the broker configuration.

object-name

The name of the database or the far sync with which the `instance-name` is associated.

Usage Notes

- The broker automatically adds started instances to the broker configuration. However, the broker does not automatically remove instances from the database. The `REMOVE INSTANCE` command can be used to manually remove any instance that no longer exists from the configuration.
- If the `instance-name` is not unique within the configuration, then you must specify both the database-name or far sync-name, and the instance-name to fully identify the instance.
- This command is rejected for an instance that is currently active in the broker configuration.
- This command is rejected if this is the only instance currently associated with a database or far sync.

Command Example

The following example shows how to remove an instance of the database.

```
DGMGRL> REMOVE INSTANCE 'south_sales3' ON DATABASE 'South_Sales';  
Removed instance "south_sales3" from the database "South_Sales"
```

REMOVE PLUGGABLE DATABASE

This command stops recovery at a target PDB and then drops the target PDB. Dropping the target PDB ends data protection for the source PDB in a DG PDB environment.

Format

```
REMOVE PLUGGABLE DATABASE pdb-name AT database-name;
```

Parameters

Parameter	Description
pdb-name	Name of the target PDB that must be removed from target database.
database-name	Name of the target database that contains the specified target PDB.

Usage Notes

- The specified target PDB must have been configured using the `ADD PLUGGABLE DATABASE` command.

Examples

Example 10-10

This example removes the target PDB named `dgpdb_sales` from the target database `cdb_newyork`. This stops the recovery process at the target PDB and removes it from the target database.

```
DGMGRL> REMOVE PLUGGABLE DATABASE dgpdb_sales AT cdb_newyork;
Removed.
```

REMOVE RECOVERY_APPLIANCE

The `REMOVE RECOVERY_APPLIANCE` command removes the specified Zero Data Loss Recovery Appliance (Recovery Appliance) from the broker configuration and terminates broker management of the Recovery Appliance.

Caution:

When you use the `REMOVE RECOVERY_APPLIANCE` command, the Recovery Appliance profile information is deleted from the broker configuration file and cannot be recovered.

Format

```
REMOVE RECOVERY_APPLIANCE object-name [ PRESERVE DESTINATIONS ];
```

Command Parameters

object name

The name of the Recovery Appliance whose profile you want to remove from the broker configuration.

Usage Notes

- By default, this command removes the corresponding broker settings of the `LOG_ARCHIVE_DEST_n` initialization parameter on the database that was shipping

redo data to the Recovery Appliance and the `LOG_ARCHIVE_CONFIG` initialization parameter on all databases in the configuration. To preserve these settings, use the `PRESERVE DESTINATIONS` option.

Command Example

The following example shows how to remove a Recovery Appliance from a Data Guard broker configuration.

```
DGMGRL> SHOW CONFIGURATION;
Configuration - DRSolution
  Protection Mode: MaxPerformance
  Members:
    North_Sales - Primary database
    South_Sales - Physical standby database
    EnterpriseRecoveryAppliance - Oracle Backup Appliance

Fast-Start Failover: DISABLED
Configuration status:
SUCCESS

DGMGRL> REMOVE RECOVERY_APPLIANCE 'EnterpriseRecoveryAppliance';
Removed Oracle Backup Appliance "EnterpriseRecoveryAppliance" from the configuration.

DGMGRL> SHOW CONFIGURATION;
Configuration - DRSolution
  Protection Mode: MaxPerformance
  Members:
    North_Sales - Primary database
    South_Sales - Physical standby database

Fast-Start Failover: DISABLED
Configuration status:
SUCCESS
```

SET ECHO

The `SET ECHO` command controls whether or not to echo commands that are issued either at the command-line prompt or from a DGMGRL script.

Format

```
SET ECHO [ON | OFF];
```

Usage Notes

- None

Command Example

```
DGMGRL> SET ECHO ON;
DGMGRL> SHOW CONFIGURATION;
SHOW CONFIGURATION;

Configuration - DRSolution

  Protection Mode: MaxPerformance
  Members:
```

```
North_Sales - Primary database
South_Sales - Physical standby database
```

```
Fast-Start Failover: DISABLED
```

```
Configuration Status:
SUCCESS
```

SET FAST_START FAILOVER TARGET

The `SET FAST_START FAILOVER TARGET` command enables you to set the fast-start failover target to the named standby database without disabling fast-start failover or modifying the fast start failover list.

Format

```
SET FAST_START FAILOVER TARGET TO database-name [NOWAIT];
```

Command Parameters

database-name

The name of the standby database that must be the new fast-start failover target. The specified fast-start failover target must be one of the databases specified in the `FastStartFailoverTarget` property of the primary database.

NOWAIT

Sets the named standby database as the fast-start failover target immediately.

Command Example

Example 1: Setting the Fast-start Failover to a Specific Standby

The following example shows how to set the fast-start failover target to the standby database named `Boston`.

```
DGMGRL> SET FAST_START FAILOVER TARGET TO Boston;
Changing fast-start failover target to 'Boston'...
Succeeded.
```

```
DGMGRL> SHOW FAST_START FAILOVER;
```

```
Fast-Start Failover: Enabled in Zero Data Loss Mode
```

```
Protection Mode: MaxAvailability
Lag Limit: 0 seconds
```

```
Threshold:          180 seconds
Ping Interval: 3000 milliseconds
Ping Retry: 0
Active Target:      Boston
Potential Targets: "Nashua, Boston"
Nashua    valid
Boston    valid
```

```

Observer:          observer-node
Lag Limit:         30 seconds (not in use)
Shutdown Primary: TRUE
Auto-reinstate:   TRUE
Observer Reconnect: (none)
Observer Override: FALSE

```

Configurable Failover Conditions

```

Health Conditions:
  Corrupted Controlfile          YES
  Corrupted Dictionary           YES
  Inaccessible Logfile           NO
  Stuck Archiver                 NO
  Datafile Write Errors          YES

```

```

Oracle Error Conditions:
(none)

```

Example 2: Using the NOWAIT Mode with Setting a Fast Start Failover Target

The following command sets the fast start failover mode to the standby database named Boston. The `NOWAIT` clause specifies that the fast-start failover target setting must be updated immediately.

```

DGMGRL> SET FAST_START FAILOVER TARGET TO Boston NOWAIT;
Fast-start failover target switch to "Boston" requested.

```

```

DGMGRL> SHOW FAST_START FAILOVER;

```

Fast-Start Failover: Enabled in Zero Data Loss Mode

```

Protection Mode:    MaxAvailability
Lag Limit:          0 seconds

Threshold:          180 seconds
Active Target:      Nashua
Potential Targets:  "Nashua, Boston"
  Nashua    valid
  Boston    valid
Observer:           observer-node
Shutdown Primary:  TRUE
Auto-reinstate:    TRUE
Observer Reconnect: (none)
Observer Override:  FALSE

```

Configurable Failover Conditions

```

Health Conditions:
  Corrupted Controlfile          YES
  Corrupted Dictionary           YES
  Inaccessible Logfile           NO
  Stuck Archiver                 NO
  Datafile Write Errors          YES

```

```

Oracle Error Conditions:
(none)

```

```
DGMGRL> SHOW FAST_START FAILOVER;

Fast-Start Failover: Enabled in Zero Data Loss Mode

Protection Mode:      MaxAvailability
Lag Limit:           0 seconds

Threshold:           180 seconds
Active Target:       Boston
Potential Targets:   "Nashua, Boston"
    Nashua    valid
    Boston    valid
Observer:            observer-node
Shutdown Primary:   TRUE
Auto-reinstate:     TRUE
Observer Reconnect: (none)
Observer Override:  FALSE

Configurable Failover Conditions
Health Conditions:
    Corrupted Controlfile      YES
    Corrupted Dictionary      YES
    Inaccessible Logfile      NO
    Stuck Archiver            NO
    Datafile Write Errors     YES

Oracle Error Conditions:
(none)
```

SET MASTEROBSERVER TO

The `SET MASTEROBSERVER TO` command lets you manually change which observer is recognized as the master observer.

Use the `SET MASTEROBSERVER TO` command to manually designate which observer is to be used as the master.

Format

```
SET MASTEROBSERVER TO observer_name
```

Command Parameters

observer_name

The name of the observer that you want to be the master observer.

Usage Notes

- If the specified observer name does not exist, an error message is returned and the master observer is not changed
- When this command is issued, the actual switch does not happen until the next time the primary contacts the target standby, usually within three seconds if fast-

start failover is enabled. You should use the `SHOW OBSERVER` command to verify that the switch took place.

- For the manual setting to succeed, the following conditions must be met during the next fast-start failover ping:
 - The target standby is enabled and does not require reinstatement.
 - There is no role change, reinstating, or fast-start failover target switch in progress

Command Example

The following is an example of designating a new observer to be the master.

```
DGMGRL> SET MASTEROBSERVER TO boston-obsever;  
Succeeded.
```

SET MASTEROBSERVERHOSTS

The `SET MASTEROBSERVERHOSTS` command sets the master observer of a broker configuration to the observer on the target host.

For each broker configuration in a specified group, if it has a backup observer running on the target host, then set the master observer of this broker configuration to the observer on the target host.

Format

```
SET MASTEROBSERVERHOSTS {FOR cfg_group_name} TO host-name;
```

Command Parameters

cfg_group_name

The name of a broker configuration group, on which you want to move the master observer to the target host.

host-name

The target host to which you want to move the master observer for the broker configurations in the specified group.

Usage Notes

- If no *cfg_group_name* is specified, then this command attempts to switch the master observer to the specified host for all broker configurations defined in the observer configuration file.
- The *cfg_group_name* cannot be the keyword `ALL`.
- The actual switch does not happen until the next time the primary contacts the target standby in each broker configuration, usually within three seconds if fast-start failover is enabled. You should use the `SHOW OBSERVERS` command to verify that the switch took place.
- Information about the DGMGRL commands run and execution details are written to the log file, `superobserver.log`. This file is located in the `$DG_ADMIN/admin/` directory. If the `DG_ADMIN` environment variable is not defined, this file is located in the current working directory.

Command Example

```
DGMGRL> SET MASTEROBSERVERHOSTS FOR GRP_A TO dgnet0;
```

SET ObserverConfigFile

The `SET ObserverConfigFile` command sets the full path and file name of an observer configuration file.

An observer configuration file stores information about managed configurations. The commands `START OBSERVING`, `STOP OBSERVING`, and `SHOW OBSERVERS` read the information about broker configuration groups from the file specified on this command.

Format

```
SET ObserverConfigFile = filename
```

Command Parameters

filename

The full path of an observer configuration file.

Usage Notes

- `ObserverConfigFile` is a DGMGRL runtime property. It neither resides in broker configuration metadata nor is persisted to disk. If the observer configuration file name is not `observer.ora` or it does not exist in the current working directory, then you must specify the name every time you start a new DGMGRL client.
- The default value of the property `ObserverConfigFile` is `observer.ora`.
- You can specify an absolute path for the observer configuration file. If you specify only a file name, the default path is the `DG_ADMIN` directory. If `DG_ADMIN` is not defined, the default path is the current working directory.
- When you issue this command, the name of the configuration file is changed even if the file you specify does not exist or the content of the file is invalid.

Command Example

```
DGMGRL> SET ObserverConfigFile = /usr/oracle/observer.ora
```

SET TIME

The DGMGRL `SET TIME` command turns timestamp printing on and off.

The timestamp printing feature records the timestamp as you input each command at the DGMGRL prompt. This information can be helpful with analysis of DGMGRL console input and output.

Format

```
SET TIME [ON | OFF];
```

Usage Notes

- None

Command Example

```
DGMGRL> SET TIME ON;  
07/11/2017 09:28:21 DGMGRL> SHOW CONFIGURATION;
```

```
Configuration - DRSolution
```

```
Protection Mode: MaxPerformance  
Members:  
North_Sales - Primary database  
South_Sales - Physical standby database
```

```
Fast-Start Failover: DISABLED
```

```
Configuration Status:  
SUCCESS
```

```
07/11/2017 09:28:24 DGMGRL> SET TIME OFF;  
DGMGRL>
```

SET TRACE_LEVEL

The `SET TRACE_LEVEL` command sets the amount of tracing done by DGMGRL. This is a client-side setting and does not impact the tracing set for the broker within the Oracle Database.

Format

```
SET TRACE_LEVEL [ SUPPORT | USER | NONE];
```

Usage Notes

- Set trace level to `USER` to limit the amount of tracing information stored. This is the default setting and includes information about fast-start failover, status changes of the primary and target standby database, and error or warning messages.
- Set trace level to `SUPPORT` to increase the amount of tracing information to include lower-level information needed by Oracle Support Services.

Example 10-11 Setting the DGMGRL and Observer Tracing Levels

The following example starts DGMGRL with the `TRACE_LEVEL` set to the default setting of `USER`. The `SET TRACE_LEVEL` command is used to modify the trace level for DGMGRL to `SUPPORT`.

When you subsequently start the observer, use the `TRACE_LEVEL` clause to set the observer's trace level to `USER`. If you omit the `TRACE_LEVEL` clause in the `START OBSERVER` command, the observer is started using the same trace level setting as `DGMGRL, SUPPORT`.

```
$ dgmgrl
DGMGRL> SET TRACE_LEVEL SUPPORT;
DGMGRL> START OBSERVER TRACE_LEVEL is USER;
```

Example 10-12 Setting the DGMGRL and Database Tracing Levels to Different Values

The following example sets the DGMGRL trace level to `SUPPORT`. The `EDIT CONFIGURATION` command is used to set the trace level of the Oracle Database to `USER`. Therefore, DGMGRL and the database use different tracing levels.

```
$ dgmgrl
DGMGRL> SET TRACE_LEVEL support;
DGMGRL> EDIT CONFIGURATION SET PROPERTY TraceLevel = USER;

=> Server-side trace level changes to USER while DGMGRL's trace level
is still SUPPORT.
```

SHOW ALL

The `SHOW ALL` command shows the values of DGMGRL CLI properties.

Format

```
SHOW ALL;
```

Usage Notes

- None

Command Example

```
DGMGRL> SHOW ALL;
debug                ON
echo                 OFF
time                 OFF
observerconfigfile = observer.ora
```

SHOW CONFIGURATION

The `SHOW CONFIGURATION` command displays a summary and status of the broker configuration.

The summary lists all members included in the broker configuration and other information pertaining to the broker configuration itself, including the fast-start failover status and the transport lag and apply lag of all standby databases.

Format

```
SHOW CONFIGURATION LAG [VERBOSE | property-name];
```

Command Parameters**LAG**

Displays the following information about the broker configuration:

- transport lag and apply lag for every standby member
- transport lag for every far sync instance

property-name

The name of the property for which you want to display summary information.

See [Oracle Data Guard Broker Properties](#) for complete information about properties.

Usage Notes

- Use the `SHOW CONFIGURATION VERBOSE` command (or the `SHOW FAST_START FAILOVER` command) to show the properties related to fast-start failover.
- You can optionally specify either `VERBOSE` or `property-name`, but not both.
- The `SHOW CONFIGURATION` command displays the status of the configuration and its members as of the last time the health was evaluated. (The health of the configuration and its members is evaluated once a minute.)

Specifying the `VERBOSE` keyword forces an immediate health evaluation of the configuration and its members before the health information is displayed.

- During a rolling upgrade done using the PL/SQL package `DBMS_ROLLING`, the `SHOW CONFIGURATION` command shows `Transient logical standby database` as the role of the upgrade target, and `ROLLING DATABASE MAINTENANCE IN PROGRESS` as the configuration status. See Example 3.
- The display highlights the current fast-start failover target with an asterisk (*) when fast-start failover is enabled

Command Examples**Example 1: Showing a Summary of the DRSolution Configuration**

The following example provides a summary of the `DRSolution` configuration for which fast-start failover is disabled. The output shows a far sync instance named `FS` in the broker configuration. The `North_Sales` database is shipping to `FS`, and `FS` is shipping to `South_Sales`.

```
DGMGRL> SHOW CONFIGURATION;

Configuration - DRSolution

  Protection Mode: MaxAvailability
  Members:
  North_Sales - Primary database
    FS- Far sync instance
    South_Sales- Physical standby database

Fast-Start Failover: DISABLED
```

```
Configuration Status:  
SUCCESS (status updated 20 seconds ago)
```

Example 2: Showing Detailed Description of the DRSolution Configuration

The following example provides detailed information about the DRSolution configuration, including configuration properties, and fast-start failover-related information:

```
DGMGRL> SHOW CONFIGURATION VERBOSE;  
  
Configuration - DRSolution  
  
Protection Mode: MaxAvailability  
Members:  
  
North_Sales - Primary database  
  FS- Far sync instance  
  South_Sales- (*) Physical standby database  
  
(* ) Fast-Start Failover target  
  
Properties:  
  FastStartFailoverThreshold      = '30'  
  OperationTimeout                = '30'  
  TraceLevel                      = 'USER'  
  FastStartFailoverLagLimit       = '30'  
  CommunicationTimeout            = '180'  
  ObserverReconnect               = '0'  
  FastStartFailoverAutoReinstate  = 'TRUE'  
  FastStartFailoverPmyShutdown    = 'TRUE'  
  BystandersFollowRoleChange      = 'ALL'  
  ObserverOverride                = 'FALSE'  
  ExternalDestination1            = ''  
  ExternalDestination2            = ''  
  PrimaryLostWriteAction          = 'CONTINUE'  
  ConfigurationWideServiceName    = 'North_Sales_CFG'  
  
Fast-Start Failover: Enabled in Zero Data Loss Mode  
  
Lag Limit:          0 seconds  
Threshold:          30 seconds  
Active Target:      South_Sales  
Potential Targets: "South_Sales"  
  South_Sales      valid  
Observer:           observer.example.com  
Shutdown Primary:  TRUE  
Auto-reinstate:    TRUE  
Observer Reconnect: (none)  
Observer Override: FALSE  
  
Configuration Status:  
WARNING
```

Example 3: Sample Output During a Rolling Upgrade Performed with the DBMS_ROLLING Package

```
Configuration - DRSolution  
  
Protection Mode: MaxPerformance
```

```

Members:

North_Sales - Primary database
South_Sales - Transient logical standby database

Fast-Start Failover: DISABLED

Configuration Status:
ROLLING DATABASE MAINTENANCE IN PROGRESS

```

Example 4: Showing Detailed Transport and Apply Lag Information

```
DGMGRL> SHOW CONFIGURATION LAG VERBOSE ;
```

```
Configuration - The SUPER cluster
```

```

Protection Mode: MaxPerformance
Members:
dgb6 - Primary database
dgb6c - Physical standby database
      Transport Lag:      0 seconds (computed 1 second ago)
      Apply Lag:          0 seconds (computed 1 second ago)
dgb6e - Far sync instance
      Transport Lag:      0 seconds (computed 1 second ago)
dgb6b - Snapshot standby database
      Transport Lag:      48 seconds (computed 2 seconds ago)
      Apply Lag:          53 seconds (computed 1 second ago)
dgb6d - Logical standby database
      Transport Lag:      0 seconds (computed 1 second ago)
      Apply Lag:          0 seconds (computed 1 second ago)

```

```

Properties:
FastStartFailoverThreshold      = '180'
OperationTimeout                 = '30'
TraceLevel                       = 'USER'
FastStartFailoverLagLimit       = '300'
CommunicationTimeout            = '180'
ObserverReconnect                = '0'
FastStartFailoverAutoReinstate  = 'TRUE'
FastStartFailoverPmyShutdown    = 'TRUE'
BystandersFollowRoleChange      = 'ALL'
ObserverOverride                 = 'FALSE'
ExternalDestination1            = ''
ExternalDestination2           = ''
PrimaryLostWriteAction          = 'CONTINUE'
ConfigurationWideServiceName    = 'b6_CFG'

```

```
Fast-Start Failover: DISABLED
```

SHOW CONFIGURATION WHEN PRIMARY IS

The `SHOW CONFIGURATION WHEN PRIMARY IS` command displays the redo transport configuration that would be in effect if the specified database were the primary database.

The display lists all members, but unlike `SHOW CONFIGURATION`, it does not provide status or any other configuration information.

Format

`SHOW CONFIGURATION WHEN PRIMARY IS database-name;`

Command Parameters

database-name

The name of the database for which you want to see what the redo transport configuration would be if it were the primary database.

Usage Notes

- Use the `SHOW CONFIGURATION WHEN PRIMARY IS` command to show the redo transport configuration that would be in effect if the specified database were the primary database. You can use this information to identify ahead of time any redo transport configurations that would be incorrect after a role change.

Command Example

The following example provides a summary of the `DRSolution` configuration before and after a role change to the `South_Sales` database.

```
DGMGRL> SHOW CONFIGURATION;

Configuration - DRSolution

Protection Mode: MaxAvailability
Members:
North_Sales - Primary database
North_FS    - Far Sync
South_Sales - Physical standby database

Members Not Receiving Redo:
South_FS - Far Sync

Fast-Start Failover: DISABLED

Configuration Status:
SUCCESS

DGMGRL> SHOW CONFIGURATION WHEN PRIMARY IS 'South_Sales';

Configuration when South_Sales is primary - DRSolution

Members:
South_Sales - Primary database
South_FS    - Far Sync
North_Sales - Physical standby database

Members Not Receiving Redo:
North_FS - Far Sync
```

These displays are based upon the `RedoRoutes` property being set as follows for each member:

```
DGMGRL> SHOW DATABASE 'North_Sales' RedoRoutes;
RedoRoutes = '(LOCAL : North_FS)'
```

```
DGMGRL> SHOW FAR_SYNC 'North_FS' RedoRoutes;
RedoRoutes = '(North_Sales : South_Sales)'
```

```
DGMGRL> SHOW DATABASE 'South_Sales' RedoRoutes;
RedoRoutes = '(LOCAL : South_FS)'
```

```
DGMGRL> SHOW FAR_SYNC 'South_FS' RedoRoutes;
RedoRoutes = '(South_Sales : North_Sales)'
```

SHOW DATABASE

The `SHOW DATABASE` command displays information, property values, or initialization parameter values of the specified database and its instances.

Format

```
SHOW DATABASE [VERBOSE] database-name [property-name] [initialization-
parameter-name];
```

Command Parameters

database-name

The name of the database for which you want to display information. The `VERBOSE` keyword, if used, must come before the *database-name* or an error is returned.

property-name

The name of the property for which you want to display a value. If a property name is specified, the output shows only the specified property (not all properties of the database), regardless of whether or not the `VERBOSE` keyword is specified.



See Also:

[Managing the Members of a Broker Configuration](#) and [Oracle Data Guard Broker Properties](#) for information about properties.

initialization-parameter-name

The name of the database initialization parameter for which you want to display a value. If a parameter name is specified, the output shows only the specified parameter (not all parameters), regardless of whether or not the `VERBOSE` keyword is specified.

Usage Notes

- The database whose properties are being set must be active.
- You must connect to the database whose property is being set by using any technique other than operating system authentication.
- The `SHOW DATABASE` command shows a brief summary of the database. The `SHOW DATABASE VERBOSE` command shows properties of the database in addition to the brief summary. They both show the status of the database.

- The `SHOW DATABASE VERBOSE` command shows the locations of the Oracle alert log file and of the broker log file. The broker log file is created in the same directory as the alert log and is named `drac<$ORACLE_SID>.log`.
- The `SHOW DATABASE VERBOSE` command shows database-specific properties and instance-specific properties. For a non-Oracle RAC database, the values of the instance-specific properties are those of the only instance of the database. For an Oracle RAC database, the values of the instance-specific properties will not be shown, although the property names are still listed. To see the instance-specific values of these properties, use the `SHOW INSTANCE` command.
- The properties that the `SHOW DATABASE VERBOSE` command shows depend on the database role and the configuration composition:
 - For the primary database, properties specific to physical or snapshot standby databases are shown only if there is at least one physical or snapshot standby database in the configuration. The properties specific to logical standby databases are shown only if there is at least one logical standby database in the configuration.
 - For physical and snapshot standby databases, properties specific to logical standby databases are not shown.
 - For logical standby databases, properties specific to physical and snapshot standby databases are not shown.
- This command is rejected if you use the `SHOW DATABASE database-name property-name` command to show an instance-specific property in an Oracle RAC database.
- During a rolling upgrade done using the PL/SQL package `DBMS_ROLLING`, the `SHOW DATABASE` command shows a `WARNING` with an appropriate `ORA` error for the upgrade target and the trailing or leading standbys, depending on the current rolling upgrade progress. See Example 3.

Command Examples

Example 1: Showing Database Information in Abbreviated Format

This example shows database information in an abbreviated format.

```
DGMGRL> SHOW DATABASE South_Sales;

Database - South_Sales

Role:                PHYSICAL STANDBY
Intended State:      APPLY-ON
Transport Lag:       (unknown)
Apply Lag:           0 seconds (computed 0 seconds ago)
Apply Rate:          1.73 MByte/s
Real Time Query:    OFF
Instance(s):
  south_sales1

Database Status:
SUCCESS
```

Example 2: Showing Database Information in Extended Format

This example shows database information in an extended format.

```
DGMGRL> SHOW DATABASE VERBOSE 'North_Sales';
```

Database - North_Sales

```
Role:                PRIMARY
Intended State:      TRANSPORT-ON
Instance(s):
  North_Sales1
```

Properties:

```
DGConnectIdentifier      = 'North_Sales.example.com'
ObserverConnectIdentifier = ''
FastStartFailoverTarget  = ''
PreferredObserverHosts   = ''
LogShipping              = 'ON'
RedoRoutes               = ''
LogXptMode               = 'ASYNC'
DelayMins                = '0'
Binding                 = 'optional'
MaxFailure               = '0'
ReopenSecs              = '300'
NetTimeout              = '30'
RedoCompression         = 'DISABLE'
PreferredApplyInstance   = ''
ApplyInstanceTimeout     = '0'
ApplyLagThreshold        = '0'
TransportLagThreshold    = '0'
TransportDisconnectedThreshold = '0'
ApplyParallel           = 'AUTO'
ApplyInstances           = '0'
ArchiveLocation          = ''
AlternateLocation        = ''
StandbyArchiveLocation   = 'USE_DB_RECOVERY_FILE_DEST'
StandbyAlternateLocation = ''
InconsistentProperties   = '(monitor)'
InconsistentLogXptProps  = '(monitor)'
LogXptStatus            = '(monitor)'
SendQEntries            = '(monitor)'
RecvQEntries            = '(monitor)'
HostName                = 'North_Sales.example.com'
StaticConnectIdentifier   = '(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)
(HOST=North_Sales.example.com) (PORT=2840))
(CONNECT_DATA=(SERVICE_NAME=North_Sales_DGMGRL.example.com)
(INSTANCE_NAME=north_sales1) (SERVER=DEDICATED)))'
TopWaitEvents            = '(monitor)'
SidName                 = '(monitor)'
```

Log file locations:

```
Alert log                : /dev/oracle/log/diag/rdbms/North_Sales/
north_sales1/trace/alert_north_sales1.log
Data Guard Broker log    : /dev/oracle/log/diag/rdbms/North_Sales/
north_sales1/trace/drcnorth_sales1.log
```

Database Status:
SUCCESS

Example 3: Sample Output for the Target Database During a Rolling Upgrade Performed With the DBMS_ROLLING Package

```
DGMGRL> SHOW DATABASE South_Sales;
```

Database - South_Sales

```
Role: Physical standby database
Intended State: APPLY-ON
Transport Lag: 0 seconds (computed 1 second ago)
Apply Lag: 0 seconds (computed 1 second ago)
Average Apply Rate: 6.00 KByte/s
Real Time Query: OFF
Instance(s):
  South
```

```
Database Warning(s):
  ORA-16866: database converted to transient logical standby
  database for rolling database maintenance
```

Database Status:
WARNING

Example 4: Sample Output For the Leading Standby During a Rolling Upgrade Performed With the DBMS_ROLLING Package

```
DGMGRL> SHOW DATABASE South_Sales;
```

Database - South_Sales

```
Role: Physical standby database
Intended State: APPLY-ON
Transport Lag: 0 seconds (computed 0 seconds ago)
Apply Lag: 0 seconds (computed 0 seconds ago)
Average Apply Rate: 510.00 KByte/s
Real Time Query: OFF
Instance(s):
  South
```

```
Database Warning(s):
  ORA-16881: standby database is not protecting the current primary
  database during rolling database maintenance
```

Database Status:
WARNING

SHOW FAR_SYNC

The `SHOW FAR_SYNC` command shows information about a far sync instance.

Format

```
SHOW FAR_SYNC [VERBOSE] far_sync_instance_name [property-name]  
[initialization-parameter-name];
```

Command Parameters

far_sync_instance_name

The name of the far sync instance for which the broker will show information. It must match (case-insensitive) the value of the corresponding database `DB_UNIQUE_NAME` initialization parameter.

property-name

The name of the property for which you want to display a value. If a property name is specified, the output shows only the specified property (not all properties of the far sync), regardless of whether or not the `VERBOSE` keyword is specified.



See Also:

[Managing the Members of a Broker Configuration and Oracle Data Guard Broker Properties](#) for information about properties.

initialization-parameter-name

The name of the database initialization parameter for which you want to display a value. If an initialization parameter name is specified, the output shows only the specified initialization parameter (not all initialization parameters of the far sync), regardless of whether or not the `VERBOSE` keyword is specified.

Command Examples

Example 1: Sample SHOW FAR_SYNC Output Without VERBOSE

The following example shows sample output from this command:

```
DGMGRL> SHOW FAR_SYNC FS;  
  
Far Sync - FS  
  
Transport Lag: 0 seconds (computed 1 second ago)  
Instance(s):  
  fs1  
  
Far Sync Status:  
SUCCESS
```

Example 2: Sample SHOW FAR SYNC Output With VERBOSE

The following example shows sample output from this command when the `VERBOSE` option is used:

```

DGMGRL> SHOW FAR_SYNC VERBOSE FS;

Far Sync - FS

Transport Lag: 0 seconds (computed 0 seconds ago)
Instance(s):
  b02

Properties:
  DGConnectIdentifier      = 'fs.example.com'
  LogXptMode               = 'sync'
  RedoRoutes               = '(North_Sales : South_Sales)
(South_Sales : North_Sales)'
  Binding                  = 'optional'
  MaxFailure                = '0'
  ReopenSecs               = '300'
  NetTimeout               = '30'
  RedoCompression         = 'DISABLE'
  LogShipping              = 'ON'
  TransportLagThreshold    = '0'
  TransportDisconnectedThreshold = '0'
  InconsistentProperties    = '(monitor)'
  InconsistentLogXptProps  = '(monitor)'
  LogXptStatus             = '(monitor)'
  HostName                 = 'fs.example.com'
  StandbyArchiveLocation   = 'USE_DB_RECOVERY_FILE_DEST'
  StandbyAlternateLocation = ''
  TopWaitEvents            = '(monitor)'
  SidName                  = '(monitor)'

Far Sync Status:
SUCCESS

```

SHOW FAST_START FAILOVER

The `SHOW FAST_START FAILOVER` command displays all fast-start failover related information.

If there is more than one registered observer running, then the output of this command shows all registered observers and indicates, with an asterisk, which one is the master observer. To see information in addition to host names of observers, use the `SHOW OBSERVER` command.

Format

```
SHOW FAST_START FAILOVER;
```

Command Parameters

None.

Usage Notes

- The `SHOW FAST_START FAILOVER` command shows a summary of the fast-start failover configuration.
- The display shows the current fast-start failover target as well as candidate fast-start failover targets. If the `FastStartFailoverTarget` property of the primary

database is set to ANY, then the candidate targets would include the standby databases that are properly configured for the prevailing protection mode.

Command Examples

Example 1: This example shows the output when there is only one registered observer running and there are multiple candidate targets.

```
DGMGRL> SHOW FAST_START FAILOVER;

Fast-Start Failover: Enabled in Zero Data Loss Mode

Protection Mode: MaxAvailability
Lag Limit: 0 seconds

Threshold: 180 seconds
Ping Interval: 3000 milliseconds
Ping Retry: 0
Active Target: South_Sales
Potential Targets: East_Sales, West_Sales
                   East_Sales    valid
                   West_Sales    valid
Observer: observer.example.com
Shutdown Primary: TRUE
Auto-reinstate: TRUE
Observer Reconnect: (none)
Observer Override: FALSE

Configurable Failover Conditions
Health Conditions:
Corrupted Controlfile YES
Corrupted Dictionary YES
Inaccessible Logfile NO
Stuck Archiver NO
Datafile Write Errors YES

Oracle Error Conditions:
(none)
```

Example 2: This example shows the output when there are multiple registered observers running. The asterisk symbol (*) indicates which observer is the master.

```
DGMGRL> SHOW FAST_START FAILOVER;

Fast-Start Failover: Enabled in Zero Data Loss Mode

Protection Mode: MaxAvailability
Lag Limit: 0 seconds

Threshold: 180 seconds
Ping Interval: 3000 milliseconds
Ping Retry: 0
Active Target: South_Sales
```

```
Potential Targets: "East_Sales, West_Sales"  
    East_Sales valid  
    West_Sales valid  
Observer: observer.example.com  
Shutdown Primary: TRUE  
Auto-reinstate: TRUE  
Observer Reconnect: (none)  
Observer Override: FALSE  
  
Configurable Failover Conditions  
Health Conditions:  
    Corrupted Controlfile YES  
    Corrupted Dictionary YES  
    Inaccessible Logfile NO  
    Stuck Archiver NO  
    Datafile Write Errors YES  
  
Oracle Error Conditions:  
    (none)
```

SHOW INSTANCE

The `SHOW INSTANCE` command displays information or property values for the specified instance.

Format

```
SHOW INSTANCE [VERBOSE] instance-name [property-name] [ON  
{DATABASE database-name | FAR_SYNC far-sync-name }];
```

Command Parameters

instance-name

The name of the instance for which you want to display information. The `VERBOSE` keyword, if used, must come before the instance name.

property-name

The name of the property for which you want to display a value. If a property name is specified, the output shows only the specified property (not all properties), regardless of whether or not the `VERBOSE` keyword is specified.



See Also:

[Managing the Members of a Broker Configuration and Oracle Data Guard Broker Properties](#) for information about properties.

database-name | far-sync-name

The name of the database or far sync associated with the instance for which you want to show information.

Usage Notes

- The `SHOW INSTANCE` command shows a brief summary of the instance. The `SHOW INSTANCE VERBOSE` command shows properties of the instance in addition to the brief summary. They both show the status of the instance.
- The `SHOW INSTANCE VERBOSE` command shows the locations of the Oracle alert log file and of the broker log file. The broker log file is created in the same directory as the alert log and is named `drc<${ORACLE_SID}>.log`.
- The `SHOW INSTANCE VERBOSE` command only shows instance-specific properties.
- The properties that the `SHOW INSTANCE VERBOSE` command shows depend on the database role and the configuration composition:
 - For instances of the primary database, properties specific to physical or snapshot standby instances are shown only if there is at least one physical or snapshot standby database in the configuration. The properties specific to logical standby instances are shown only if there is at least one logical standby database in the configuration.
 - For instances of physical or snapshot standby databases, properties specific to logical standby instances are not shown.
 - For instances of logical standby databases, properties specific to physical and snapshot standby instances are not shown.
- The *instance-name* can be unique across the configuration. If *instance-name* is not unique, you must specify both the *database-name* and the *instance-name* to fully identify the instance.

Command Examples

Example 1: Showing Instance Information in Abbreviated Format

The following example shows information about a specific instance of a database.

```
DGMGRL> SHOW INSTANCE 'north_sales1';

Instance 'north_sales1' of database 'North_Sales'

Instance Status:
SUCCESS
```

Example 2: Showing Instance Information in Extended Format

The following example shows instance information in an extended format.

```
DGMGRL> SHOW INSTANCE VERBOSE 'north_sales1';

Instance 'north_sales1' of database 'North_Sales'

PFILE:
Properties:
  HostName                = 'north.example.com'
  StaticConnectIdentifier  = '(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)
(HOST=north.example.com) (PORT=2094))
(CONNECT_DATA=(SERVICE_NAME=North_Sales_DGMGRL.example.com)
(INSTANCE_NAME=north_sales1) (SERVER=DEDICATED)))'
  TopWaitEvents            = '(monitor)'
  SidName                  = '(monitor)'
```

```
Log file locations:
  Alert log           : /db/oracle/log/diag/rdbms/North_Sales/north_sales1/
trace/alert_north_sales1.log
  Data Guard Broker log : /db/oracle/log/diag/rdbms/North_Sales/north_sales1/
trace/drcnorth_sales1.log

Instance Status:
SUCCESS
```

SHOW OBSERVER

The `SHOW OBSERVER` command shows information about all registered observers in a Data Guard broker configuration.

The `SHOW OBSERVER` command displays the observer name, the host name where the observer is running, whether the observer is the master observer, and the last time the observer pinged the primary and standby. It shows this information for all the observers (up to 3) in this single configuration.

FORMAT

```
SHOW OBSERVER;
```

Command Parameters

None

Usage Notes

- This command requires a DGMGRL session, which submits this command, to be connected to a single configuration.

Command Example

The following example `SHOW OBSERVER` command displays information about all registered observers in the `DRSolution` broker configuration.

```
DGMGRL> SHOW OBSERVER;

Configuration - DRSolution

Primary:           North_Sales
Active Target Standby: South_Sales

Observer "ob2" - Master

  Host Name:           observer2.example.com
  Last Ping to Primary: 1 second ago
  Last Ping to Target:  2 seconds ago

Observer "ob1" - Backup

  Host Name:           observer1.example.com
  Last Ping to Primary: 1 second ago
  Last Ping to Target:  3 seconds ago

Observer "ob3" - Backup
```

```
Host Name:                observer3.example.com
Last Ping to Primary:    4 seconds ago
Last Ping to Target:    5 seconds ago
```

SHOW ObserverConfigFile

The `SHOW ObserverConfigFile` command shows the value of the `ObserverConfigFile` property.

Format

```
SHOW ObserverConfigFile;
```

Command Parameters

None.

Usage Notes

- If the value of the `ObserverConfigFile` property is an empty string, then the output is `current_working_directory/observer.ora`.
- The `SHOW ObserverConfigFile` command attempts to parse the file pointed by the `ObserverConfigFile` property. If the file does not exist or parsing fails, then a message is returned that the file is not usable.

Command Example

```
DGMGRL> SHOW ObserverConfigFile;
ObserverConfigFile=/usr/oracle/observer
observer configuration file parsing succeeded
```

SHOW OBSERVERS

The `SHOW OBSERVERS` command shows information about all observers for all broker configurations in a specific configuration group.

The `SHOW OBSERVERS` command displays the observer name, the host name where the observer is running, whether the observer is the master observer, and the last time the observer pinged the primary and standby.

Format

```
SHOW OBSERVERS [FOR cfg_group_name];
```

Command Parameters

cfg_group_name

The name of a valid broker configuration group file for which you want to show information about all running observers. Specifying this parameter results in information being shown about observers for all configurations in the specified group. The information shown by this command is the same as that shown by a `SHOW OBSERVER` command on an individual configuration.

If a group name is not specified, then `SHOW OBSERVERS` alone is also a valid command. It shows observer information for all broker configuration groups defined in the observer configuration file. The configuration group name cannot be `ALL`.

Usage Notes

- This command can be used to verify that a manually performed switch to a new master observer was successful.

Command Example

```
DGMGRL> SHOW OBSERVERS;
ObserverConfigFile=observer.ora
observer configuration file parsing succeeded
Submit command SHOW OBSERVER using the connect identifier
'North_Sales'.
Connected to "North_Sales"

Configuration - DrSolution1

Primary:           North_Sales
Target:            South_Sales

Observer DRSolution1_Observer - Master

Host Name:         observer1.example.com
Last Ping to Primary: 3 seconds ago
Last Ping to Target: 3 seconds ago

Submit command SHOW OBSERVER using the connect identifier 'East_Sales'.
Connected to "East_Sales"

Configuration - DRSolution2

Primary:           East_Sales
Target:            West_Sales

Observer DRSolution2_Observer - Master

Host Name:         observer2.example.com
Last Ping to Primary: 3 seconds ago
Last Ping to Target: 3 seconds ago
```

SHOW PLUGGABLE DATABASE

This command displays the information or property values of the specified pluggable database (PDB).

Prerequisites

- Connect to the source PDB or target PDB as the `DGPDB_INT` user.

Format

```
SHOW PLUGGABLE DATABASE pdb-name AT database-name [ALL];
```

Parameters

Parameter	Description
<i>pdb-name</i>	Name of the PDB whose details must be displayed.
<i>database-name</i>	Name of the database that contains the PDB specified in <i>pdb-name</i> . Include the ALL keyword to display information about all source PDBs and target PDBs at the database specified by <i>database-name</i> .

Examples

Example 10-13 Displaying Details of a PDB

This example shows details about the PDB *sales* in an abbreviated format.

```
DGMGRL> SHOW PLUGGABLE DATABASE 'sales' AT boston;
Pluggable database 'sales' at database 'boston'
  Data Guard Role: Physical Standby
  Con_ID: 7
  Source: con_id 6 at newyork
  Transport Lag: +00 00:00:00 seconds
  Intended State: APPLY-ON
  Apply State: Running
  Apply Instance: boston
  Average Apply Rate: 16 KByte/s
  Real Time Query: OFF
```

SHOW RECOVERY_APPLIANCE

The `SHOW RECOVERY_APPLIANCE` command displays information, property values, or initialization parameter values of the specified Zero Data Loss Recovery Appliance (Recovery Appliance).

Format

```
SHOW RECOVERY_APPLIANCE [VERBOSE] object name [property name]
[initialization parameter name];
```

Command Parameters

object name

The name of the Recovery Appliance for which you want to display information. The `VERBOSE` keyword, if used, must come before the Recovery Appliance name or an error is returned.

property name

The name of the property for which you want to display a value. If a property name is specified, the output shows only the specified property (not all properties of the Recovery Appliance), regardless of whether or not the `VERBOSE` keyword is specified.

initialization parameter name

The name of the database initialization parameter for which you want to display a value. If a initialization parameter name is specified, the output shows only the specified initialization parameter (not all initialization parameter of the Recovery Appliance), regardless of whether or not the `VERBOSE` keyword is specified.

Usage Notes

- The `SHOW RECOVERY_APPLIANCE` command shows a brief summary of the Recovery Appliance. The `SHOW RECOVERY_APPLIANCE VERBOSE` command shows properties of the Recovery Appliance in addition to the brief summary. They both show the status of the database.
- The `SHOW RECOVERY_APPLIANCE VERBOSE` command shows Recovery Appliance-specific properties.

Command Examples**Example 1: Recovery Appliance Information in Abbreviated Format**

The following example shows Recovery Appliance information in an abbreviated format.

```
DGMGR> SHOW RECOVERY_APPLIANCE 'EnterpriseRecoveryAppliance';
Oracle Recovery Appliance - EnterpriseRecoveryAppliance
  Transport Lag:    0 seconds
  Redo Source:     South_Sales

Oracle Backup Appliance Status:
SUCCESS
```

Example 2: Recovery Appliance Information in Extended Format

The following example uses the `VERBOSE` parameter to show Recovery Appliance information in an extended format.

```
DGMGR> SHOW RECOVERY_APPLIANCE VERBOSE 'EnterpriseRecoveryAppliance';

Oracle Backup Appliance - EnterpriseRecoveryAppliance

  Transport Lag:    0
  Redo Source:     South_Sales

Properties:
  DGConnectIdentifier      = 'South_Sales.example.com'
  LogXptMode              = 'ASYNC'
  DelayMins               = '0'
  Binding                 = 'optional'
  MaxFailure              = '0'
  ReopenSecs              = '300'
  NetTimeout              = '30'
  RedoCompression        = 'DISABLE'
  LogShipping             = 'ON'
  InconsistentProperties  = '(monitor)'
  InconsistentLogXptProps = '(monitor)'
```

```
StandbyAlternateLocation      = ''  
RedoSource                    = 'South_Sales'
```

```
Database Status:  
SUCCESS
```

SHUTDOWN

Shuts down a currently running Oracle instance.

Format

```
SHUTDOWN [ ABORT | IMMEDIATE | NORMAL ];
```

Command Parameters

None.

Usage Notes

- Using the `SHUTDOWN` command with no arguments is equivalent to using the `SHUTDOWN NORMAL` command.
- The following list describes the options to the `SHUTDOWN` command:

- `ABORT`

Proceeds with the fastest possible shutdown of the database without waiting for calls to complete or for users to disconnect from the database. Uncommitted transactions are not rolled back. Client SQL statements being processed are terminated. All users connected to the database are implicitly disconnected, and the next database startup will require instance recovery. You must use this option if a background process terminates abnormally.

 **Caution:**

If you use the `ABORT` option on the primary database when fast-start failover is enabled and the observer is running, a fast-start failover may ensue. Use the `IMMEDIATE` or `NORMAL` option to prevent an unexpected fast-start failover from occurring.

- `IMMEDIATE`

Does not wait for current calls to complete or users to disconnect from the database. Further connections are prohibited. The database is closed and dismounted. The instance is shut down, and no instance recovery is required on the next database startup.

- `NORMAL`

This is the default option. The process waits for users to disconnect from the database. Further connections are prohibited. The database is closed and dismounted. The instance is shut down, and no instance recovery is required on the next database startup.

Command Example

The following command shuts down the primary database in normal mode.

```
DGMGRL> SHUTDOWN;  
  
Database closed.  
Database dismounted.  
Oracle instance shut down.
```

SPOOL

The `SPOOL` command records the input and output of DGMGRL to a file.

Format

The `SPOOL` command has three possible formats:

```
SPOOL;  
  
SPOOL spool-file-name [CREATE | REPLACE | APPEND];  
  
SPOOL OFF;
```

If you simply enter `SPOOL` at the DGMGRL command prompt, then the current spool file name is displayed.

Otherwise, the available spooling options are defined as follows:

- `CREATE`—Create a new log file. If a spool file with the specified name already exists, the `SPOOL` command fails.
- `REPLACE`—Replace the existing spool file of the name specified. This is the default behavior if no option is specified.
- `APPEND`—Append the new log into the specified log file, if it exists. Otherwise create a new one.
- `OFF`—Turns spooling off.

Command Parameters

spool-file-name

The name of the file to which DGMGRL input and output will be written.

Usage Notes

- None

Command Example

The following example shows the output of the `SPOOL` command before and after spooling is started.

```
DGMGRL> SPOOL;  
not spooling currently  
  
DGMGRL> SPOOL mysession;
```

```
DGMGRL> SPOOL;
currently spooling to "mysession"

DGMGRL>
```

SQL

The SQL command executes a SQL statement or a PL/SQL stored procedure.

Format

```
SQL "sql_statement"
```

Command Parameters

sql_statement

The SQL statement or PL/SQL stored procedure to be executed.

Usage Notes

- The SQL statement or PL/SQL stored procedure to be executed must be enclosed in double quotation marks.
- Do not include a semi-colon (;) after the SQL statement to be executed.
- If the string that DGMGRL passes to PL/SQL contains a filename, then the filename must be enclosed in single quotation marks and the entire string following the SQL command must be enclosed in double quotation marks. For example, use the following syntax:

```
SQL "CREATE TABLESPACE temp1 DATAFILE '?/oradata/trgt/temp1.dbf' SIZE 10M
TEMPORARY"
```

- You cannot execute `SELECT` statements.

Command Example

The following command opens the database.

```
DGMGRL> SQL "alter database open"
```

START OBSERVER

The `START OBSERVER` command starts a fast-start failover observer on this host (where the DGMGRL session is running), if there is no registered observer running on this host for this configuration.

Before using this command, you must first issue a `CONNECT` command to log into a specific broker configuration. Otherwise, an error message is returned stating that you are not logged on.

Format

```
START OBSERVER [observer-name] [FILE IS observer-file] [LOGFILE IS logfile]
[TRACE LEVEL IS USER | SUPPORT];
```

Command Parameters

observer-name

A name to identify observers within the same Data Guard broker configuration.

- No two observers on the same Data Guard broker configuration can have the same name.
- If no name is specified for the observer then a default observer name, the host name of machine where the `START OBSERVER` command is issued, is used.
- An observer name is case-insensitive.
- The strings "NONAME" and "ALL" cannot be used as an observer name.

observer-file

Specifies the path and name of the runtime data file. If you specify only a file name, the path used is `$DG_ADMIN/config_ConfigurationSimpleName/dat` . If you omit both the path and file name, the file name defaults to `fsfo_hostname.dat` and the path is `$DG_ADMIN/config_ConfigurationSimpleName/dat`. If the `DG_ADMIN` environment variable is not defined, the default path is the current working directory.

log-file

The full path name of the observer log file. Each observer has its own log file.

Usage Notes

- You can register up to four observers to monitor a single Data Guard broker configuration. Each observer is identified by a name that you supply when you issue the `START OBSERVER` command. See [Installing and Starting the Observer](#).
- The optional clause `TRACE_LEVEL IS` lets you control the amount of tracing done and written to the observer log file. The default value is `USER`, which limits the observer log contents to tracing information about fast-start failover, status changes of the primary and target standby database, and error/warning messages. Setting `TRACE_LEVEL` to `SUPPORT` increases the amount of tracing information to include lower-level information needed by Oracle Support Services.
- The Oracle Client Administrator kit, or the full Oracle Database Enterprise Edition or Oracle Personal Edition kit must be installed on the observer computer to monitor a broker configuration for which fast-start failover is to be enabled. See [Prerequisites for Enabling Fast-Start Failover](#) for more information.
- The `START OBSERVER` command must be issued on the observer computer. Once the observer is successfully started, control is not returned to the user until the observer is stopped (for example, by issuing the [STOP OBSERVER](#) command from a different client connection). If you want to perform further interaction with the broker configuration, you must connect through another client.

For information about how to start the observer in the background, see [START OBSERVER IN BACKGROUND](#).

- If the `LOGFILE IS` clause is used, then all observer output is recorded in the specified file. Observer output is useful for troubleshooting problems with the observer and with fast-start failover in general.

If a complete path, with file name, is provided, the file is stored in the specified path.

If only a file name is provided and the `DG_ADMIN` environment variable is defined, the specified file is stored in the `$DG_ADMIN/config_ConfigurationSimpleName/log` directory. If the `DG_ADMIN` environment variable is not defined, the file is stored in the current working directory.

If `LOGFILE IS` clause is omitted, the log file is stored in the `$DG_ADMIN/config_ConfigurationSimpleName/log` directory using the name `observer_hostname.log`. If the `DG_ADMIN` environment variable is not defined, the log file is stored as `observer_hostname.log` in the current working directory. `ConfigurationSimpleName` is the name of the broker configuration.

If the specified log file is not accessible, the observer output is sent to standard output.

- If a complete directory path and file name is specified with the `FILE IS` clause, the observer runtime data file is created in this directory. If a relative path and file name is specified, the file is created in the specified path under the current working directory.

If only a file name is specified, the file is stored in the `$DG_ADMIN/config_ConfigurationSimpleName/dat/` directory. If the `DG_ADMIN` environment variable is not defined, the file is stored in the current working directory. `ConfigurationSimpleName`, which is a configuration property, is the name of the broker configuration.

If this clause is omitted, the file is stored as `$DG_ADMIN/config_ConfigurationSimpleName/dat/FSFO_hostname.dat`. If the `DG_ADMIN` environment variable is not defined, the file is stored in the current working directory as `fsfo.dat`.

- The primary and target standby database `DB_UNIQUE_NAME` initialization parameter and connect identifiers are stored in the `fsfo.dat` configuration file. Oracle recommends you ensure this file is protected from unauthorized access.
- The order of the `FILE IS`, `LOGFILE IS`, and `TRACE LEVEL IS` clauses is interchangeable.
- Fast-start failover does not need to be enabled before you issue this command.
 - If fast-start failover is enabled, the observer will retrieve primary and target standby connect identifiers from the broker configuration and begin monitoring the configuration.
 - If fast-start failover is not enabled, the observer continually monitors for when fast-start failover is enabled.
- Only the primary database needs to be running when you issue this command; the standby database that will be the target of a fast-start failover does not need to be running in order for this command to complete successfully.
- Use the `SHOW OBSERVER` command or the `SHOW CONFIGURATION VERBOSE` command, or query the `V$FS_FAILOVER_OBSERVERS` view on the primary database to see the status of the observer and its host computer.
- If the primary and target standby databases stay connected but they lose the connection to the observer, then the primary database goes into an unobserved state. This state is reported by the broker's health check capability.
- The `SHOW OBSERVER` command indicates whether one or more observers have already been started.

If the `SHOW OBSERVER` command shows one or more registered observers, but some of them are no longer running for some reason, then you can do either of the following:

- Issue the `START OBSERVER` command on the same observer computer where it was started originally, with the observer configuration file used when the observer was first started.
- Issue the `STOP OBSERVER` command and then the `START OBSERVER` command on any computer to start the observer.

If the `SHOW OBSERVER` command shows one or more observers and one observer is already running at one location, then an attempt to start an observer at that location again will fail with the following error:

```
Unable to open the observer file
```

If the `SHOW OBSERVER` command shows four registered observers and you attempt to start an observer at a different location, then the command will fail with the following error:

```
ORA-16647: could not start more than four observers
```

Command Examples

Example 1: Starting the Observer

The following example shows how to start the observer.

```
DGMGRL> CONNECT sysdg@North_Sales.example.com;
Password: password
Connected to "North_Sales"
Connected as SYSDBG.
DGMGRL> START OBSERVER;
Observer started
```

Example 2: Starting the Observer Without Showing Credentials

The following example shows how to start the observer using `CONNECT '/'` so that connection credentials are not visible on the command line:

```
DGMGRL> CONNECT /@North_Sales.example.com;
Connected to "North_Sales"
DGMGRL> START OBSERVER;
Observer started.
```

You must set up Oracle Wallet or SSL to use `CONNECT '/'`. By setting up Oracle Wallet or SSL, you can write a script to securely start and run the observer as a background job without specifying database credentials in the script. When using Oracle Wallet as a secure external password store, be sure to add credentials for both the primary and fast-start failover target standby databases. The database connect string that you specify when adding the credentials for each database must match the `ObserverConnectIdentifier` or `DGConnectIdentifier` database property.

START OBSERVER IN BACKGROUND

The `START OBSERVER IN BACKGROUND` command starts a fast-start failover observer on this host (where this DGMGRL session is running) as a background process.

After this command is issued, DGMGRL reports whether the `START OBSERVER IN BACKGROUND` command submitted successfully. If yes, then control returns to the user. (This is different behavior from the `START OBSERVER` command, in which control does not return to the user after the observer is started.)

This command uses Oracle wallet to obtain credentials to log into the database server and register observers. Even if you have successfully connected to a database server in the broker configuration using the `CONNECT` command, this command ignores the existing connection and uses the credentials stored in Oracle wallet. If the wallet is not configured, then the command will fail to start the Observer.

Format

```
START OBSERVER [observer-name] IN BACKGROUND CONNECT IDENTIFIER IS  
connect-identifier [FILE IS observer-file] [LOGFILE IS log-file]  
[TRACE_LEVEL IS USER | SUPPORT];
```

Command Parameters

observer-name

The name to identify observers within the same data guard broker configuration.

- No two observers on the same Data Guard Broker configuration can have the same name.
- If no name is specified for the observer then a default observer name, the host name of machine where the `START OBSERVER` command is issued, is used.
- An observer name is case-insensitive.
- The string "NONAME" cannot be used as an observer name.

connect-identifier

The connect identifier for a database in the broker configuration that the fast-start failover observer will monitor. Oracle wallet uses this identifier as the key to search oracle wallet.

observer-file

Specifies the path and name of the runtime data file. If not specified, then the file name defaults to `fsfo.dat` and the path is the current working directory.

log-file

The full path of the observer log file. Each observer has its own log file.

Usage Notes

- Even if the `START OBSERVER` command is submitted successfully, the observer might fail to start due to credential problems, intermittent network connections, or failure on observer registration. To verify that the observer started successfully, use the `SHOW OBSERVERS` command or check the observer log file.
- This command ignores any connections you have made to a specific configuration member using the `CONNECT` command. In other words, even if you have not connected to a specific member in the broker configuration, you can still start an observer by using the `START OBSERVER IN BACKGROUND` command.
- If you have connected to a specific configuration member before issuing the `START OBSERVER IN BACKGROUND` command, then you can continue to use the connection after the control is returned.
- If the `observer-file` parameter is not specified with the `FILE IS` parameter, then the observer searches the current working directory for the `fsfo.dat` file. If it is not found, then the observer creates a `fsfo.dat` file.

- For the `LOGFILE IS` clause, if a complete path, with file name, is provided, the file is stored in the specified path.

If only a file name is provided and the `DG_ADMIN` environment variable is defined, the specified file is stored in the `$DG_ADMIN/config_ConfigurationSimpleName/log` directory. If the `DG_ADMIN` environment variable is not defined, the file is stored in the current working directory.

If `LOGFILE IS` clause is omitted, the log file is stored in the `$DG_ADMIN/config_ConfigurationSimpleName/log` directory using the name `observer_hostname.log`. If the `DG_ADMIN` environment variable is not defined, the log file is stored as `observer_hostname.log` in the current working directory. `ConfigurationSimpleName` is the name of the broker configuration.

- If a complete directory path and file name is specified with the `FILE IS` clause, the observer runtime data file is created in this directory. If a relative path and file name is specified, the file is created in the specified path under the current working directory.

If only a file name is specified, the file is stored in the `$DG_ADMIN/config_ConfigurationSimpleName/dat/` directory. If the `DG_ADMIN` environment variable is not defined, the file is stored in the current working directory. `ConfigurationSimpleName`, which is a configuration property, is the name of the broker configuration.

If this clause is omitted, the file is stored as `$DG_ADMIN/config_ConfigurationSimpleName/dat/FSFO_hostname.dat`. If the `DG_ADMIN` environment variable is not defined, the file is stored in the current working directory as `fsfo.dat`.

- The optional clause `TRACE_LEVEL IS` lets you control the amount of tracing done and written to the observer log file. The default value is `USER`, which limits the observer log contents to tracing information about fast-start failover, status changes of the primary and target standby database, and error/warning messages. Setting `TRACE_LEVEL` to `SUPPORT` increases the amount of tracing information to include lower-level information needed by Oracle Support Services.
- The order of the optional clauses in the `START OBSERVER IN BACKGROUND` command is interchangeable.

Command Example

```
DGMGRL> START OBSERVER observer1 IN BACKGROUND
FILE IS /net/sales/dat/oracle/broker/fsfo.dat
LOGFILE IS /net/sales/dat/oracle/broker/observer.log
CONNECT IDENTIFIER IS sales_p;
Submitted command "START OBSERVER" using connect identifier "sales_p"
```

START OBSERVING

The `START OBSERVING` command starts a new observer for each broker configuration in the specified group.

The effect of the `START OBSERVING` command is equivalent to submitting a `START OBSERVER` command on each individual configuration.

Format

```
START OBSERVING [cfg_group_name];
```

Command Parameters

cfg_group_name

The name of a broker configuration group, in which you want to start one observer for each broker configuration.

Usage Notes

- If no *cfg_group_name* is specified, then this command will start a new observer for each configuration defined in the observer configuration file.
- The *cfg_group_name* cannot be the keyword ALL.
- Information about the DGMGRL commands run and execution details are written to the log file, `superobserver.log`. This file is located in the `$DG_ADMIN/admin/` directory. If the `DG_ADMIN` environment variable is not defined, this file is located in the current working directory.

Command Example

```
DGMGRL> START OBSERVING;
ObserverConfigFile=/net/oracle/dataguard/observer.ora
observer configuration file parsing succeeded
Submitted command "START OBSERVER" using connect identifier "cfg1_cid".
Submitted command "START OBSERVER" using connect identifier "cfg2_cid".
Submitted command "START OBSERVER" using connect identifier "cfg3_cid".

Check superobserver.log and individual observer logs for execution details.
```

```
DGMGRL> START OBSERVING GRP_A;
ObserverConfigFile=/net/oracle/dataguard/observer.ora
observer configuration file parsing succeeded
Submitted command "START OBSERVER" using connect identifier "cfg1_cid".
Submitted command "START OBSERVER" using connect identifier "cfg2_cid".

Check superobserver.log and individual observer logs for execution details.
```

STARTUP

The `STARTUP` command starts an Oracle database instance, and allows you to specify a number of options.

The options you can specify are as follows:

- `FORCE`: shuts down the current Oracle instance in the `SHUTDOWN ABORT` mode before restarting it.
- `RESTRICT`: allows only Oracle users with the `RESTRICTED SESSION` system privilege to connect to the instance.
- `PFILE`: specifies the `PFILE` initialization parameter file to be used when the database instance is started.

- **MOUNT:** mounts the specified database without opening it.
- **OPEN:** mounts and opens the specified database.
- **NOMOUNT:** starts the specified database instance without mounting the database.

Format

```
STARTUP [FORCE] [RESTRICT] [PFILE=filename] [MOUNT | OPEN  
[open_options] | NOMOUNT];
```

Command Parameters

filename

The name of the initialization parameter file to be used when starting the database instance. If you do not specify the `PFILE` parameter option, then the default server parameter file (specific to your operating system) is used.

open-options

The mode of access in which you want the specified database to start. The possible modes are:

```
READ ONLY  
READ WRITE
```

Usage Notes

- Using the `STARTUP` command with no arguments is equivalent to using the `STARTUP OPEN` command.
- If you do not use the `FORCE` clause when you use the `STARTUP` command and the current database instance is running, an error results. The `FORCE` clause is useful when you are debugging or when error conditions are occurring. Otherwise, it should not be used.
- Use the `RESTRICT` clause to allow only Oracle users with the `RESTRICTED SESSION` system privilege to connect to the instance. Later, you can use the `ALTER SYSTEM` command through SQL*Plus to disable the restricted session feature.
- If you do not use the `PFILE` clause to specify the initialization parameter file, the `STARTUP` command uses the default server parameter file, if it exists. Otherwise, the `STARTUP` command uses the default initialization parameter file. The default files are platform specific.

See your operating system-specific documentation for more information about the default parameter files.
- Use the `OPEN` clause to mount and open the specified database.
- The `NOMOUNT` clause starts the database instance without mounting the database. You cannot use the `NOMOUNT` clause with the `MOUNT` or `OPEN` options.
- The order of the optional clauses in the `STARTUP` command is interchangeable.

Command Examples

Example 1: Two Methods for Starting a Database Instance

The following examples show two different methods for starting a database instance. Each command starts a database instance using the standard parameter file, mounts the default database in exclusive mode, and opens the database.

```
DGMGRL> STARTUP;  
DGMGRL> STARTUP OPEN;
```

Example 2: Shutting Down the Current Instance and Restarting Without Mounting or Opening It

The following command shuts down the current instance, immediately restarts it without mounting or opening the database, and allows only users with restricted session privileges to connect to it.

```
DGMGRL> STARTUP FORCE RESTRICT NOMOUNT;
```

Example 3: Starting (But Not Mounting) an Instance Using a Parameter File

The following command starts an instance using the parameter file `testparm` without mounting the database.

```
DGMGRL> STARTUP PFILE=testparm NOMOUNT;
```

Example 4: Starting and Mounting an Instance Without Opening It

The following example starts and mounts a database instance, but does not open it.

```
DGMGRL> STARTUP MOUNT;
```

STOP OBSERVER

The `STOP OBSERVER` command stops the fast-start failover observer.

Format

```
STOP OBSERVER [observer-name | ALL];
```

Command Parameters

observer-name

The name of the observer you want to stop. If a name is not specified and there is only one registered observer for the configuration, then it will be stopped; if there is more than one registered observer in the configuration, then an error message is returned.

The `ALL` keyword stops all observers registered in this broker configuration.

Usage Notes

- You can issue this command while connected to any database in the broker configuration.
- This command does not disable fast-start failover, but a fast-start failover cannot be initiated in the absence of an observer.
- Fast-start failover does not need to be enabled when you issue this command.
- If fast-start failover *is enabled* when you issue the `STOP OBSERVER` command, then the primary and standby databases must be connected and communicating with each other. Otherwise the following error is returned:

```
ORA-16636 fast-start failover target standby in error state, cannot stop
observer
```

If connectivity does not exist between the primary and standby databases, you can issue the `DISABLE FAST_START FAILOVER FORCE` command on the primary database and then issue the `STOP OBSERVER` command. Note that disabling fast-start failover with the `FORCE` option on a primary database that is disconnected from the observer and the target standby database does not prevent the observer from initiating a fast-start failover to the target standby database.

- If fast-start failover *is not enabled* when you issue the `STOP OBSERVER` command, then only the primary database must be running when you stop the observer.
- The observer does not stop immediately when the `STOP OBSERVER` command is issued. The observer does not discover it has been stopped until the next time the observer contacts the broker.

As soon as you have issued the `STOP OBSERVER` command, you may enter the `START OBSERVER` command again on any computer. You can start a new observer right away, even if the old observer has not yet discovered it was stopped. Any attempt to restart the old observer will fail, because a new observer has been started for the broker configuration.

- The `STOP OBSERVER` command fails if a switch to a new fast-start failover target or new master observer is underway.
- The `STOP OBSERVER` command fails if there are two or more registered observers and you attempt to stop only the master.

Command Example

The following example stops all observers running in the broker configuration .

```
DGMGRL> STOP OBSERVER ALL;
```

STOP OBSERVING

The `STOP OBSERVING` command stops all local observers running on this host (where this DGMGRL session is running) for all broker configurations in a specific group.

Format

```
STOP OBSERVING [cfg_group_name] [TRACE_LEVEL= USER | SUPPORT];
```

Command Parameters

cfg_group_name

The name of a broker configuration group, on which you want to stop local observers running on this host (where DGMGRL is running).

TRACE_LEVEL

The optional clause `TRACE_LEVEL IS` lets you control the amount of tracing done and written to the observer log file. The default value is `USER`, which limits the observer log contents to tracing information about fast-start failover, status changes of the primary and target standby database, and error/warning messages. Setting `TRACE_LEVEL` to `SUPPORT` increases the amount of tracing information to include lower-level information needed by Oracle Support Services.

Usage Notes

- If no *cfg_group_name* is specified, then this command stops all LOCAL observers running on this host (where this DGMGRL session is running) for all broker configurations defined in the observer configuration file.
- The *cfg_group_name* cannot be the keyword ALL.
- Information about the DGMGRL commands run and execution details are written to the log file, *superobserver.log*. This file is located in the *\$DG_ADMIN/admin/* directory. If the *DG_ADMIN* environment variable is not defined, this file is located in the current working directory.

Command Example

```
DGMGRL> STOP OBSERVING;
ObserverConfigFile=/net/oracle/dataguard/observer.ora
observer configuration file parsing succeeded
Submitted command "STOP OBSERVER HOST IS OBM1" using connect identifier
cfg1_cid.
Submitted command "STOP OBSERVER HOST IS OBM1" using connect identifier
cfg2_cid.
Submitted command "STOP OBSERVER HOST IS OBM1" using connect identifier
cfg3_cid.
```

Check *superobserver.log* and individual observer logs for execution details.

```
DGMGRL> STOP OBSERVING GRP_A;
ObserverConfigFile=/net/oracle/dataguard/observer.ora
observer configuration file parsing succeeded
Submitted command "STOP OBSERVER HOST IS OBM1" using connect identifier
cfg1_cid.
Submitted command "STOP OBSERVER HOST IS OBM1" using connect identifier
cfg2_cid.
```

Check *superobserver.log* and individual observer logs for execution details.

SWITCHOVER

When you issue the `SWITCHOVER` command, the current primary database becomes a standby database, and the specified standby database becomes the primary database. This is known as a switchover operation.

Format

```
SWITCHOVER TO database_name [WAIT [timeout_in_seconds]];
```

The `WAIT` option specifies that you want to wait for sessions to drain before proceeding with the switchover. Use the *timeout_in_seconds* option to specify the wait time. The broker waits for the number of seconds specified, for sessions to drain, and then proceeds with the switchover. Any sessions that have not drained will be killed during the switchover process.

If you include the `WAIT` option, but omit *timeout_in_seconds*, the broker determines the maximum `drain-timeout` value for all currently active services, waits for up to that amount of

time for all current client requests to be processed, and then proceeds with the switchover. The `drain-timeout` value is an option that is specified on the SRVCTL utility's `add service` or `modify service` commands.

The value specified for `timeout_in_seconds` overrides the value set using the `drain_timeout` option.

The `WAIT` option is valid only when services are configured with attributes related to Application Continuity in Oracle Clusterware.

Command Parameters

database-name

The name of the standby database you want to change to the primary database role.

timeout-in-seconds

The time allowed for resource draining to be completed, in seconds, before the switchover operation proceeds.

Permitted values are 0 (zero) or any positive integer. The default value is an empty string indicating that this option is not set. If the value is zero, then draining occurs immediately and broker proceeds with the switchover. During the draining period, all current client requests are processed, but new requests are not accepted.

Usage Notes

- If fast-start failover is enabled, you may switch over only to the fast-start failover target standby database.
- The broker verifies that the primary and standby databases are in the following states before starting the switchover:
 - The primary database must be enabled and in the `TRANSPORT-ON` state so redo transport services are started.
 - The standby database must be enabled and in the `APPLY-ON` state, with log apply services started.
- The broker allows the switchover to proceed as long as there are no redo transport services errors for the standby database that you selected to participate in the switchover. However, errors occurring for any other bystander standby database will not prevent the switchover from proceeding.
- Switchover to a logical standby database is not allowed when the configuration is operating in maximum protection mode.
- If the broker configuration is operating in either maximum protection mode or maximum availability mode, the switchover maintains the protection mode even after the operation (described in [Before You Perform a Switchover Operation](#)). The switchover will not be allowed if the mode cannot be maintained because the target standby database of the switchover was the only standby that satisfied the protection mode requirement.
- If the standby database that is assuming the primary role is a physical standby database, then the old primary database will be restarted after the switchover completes. Any client connections to the old primary are redirected to the physical standby database as shown in [Example 10-16](#). If the standby database is a logical standby database, then neither the primary database nor the logical standby database is restarted.

- If the standby database that is assuming the primary role is a physical standby database, then the original primary becomes a physical standby database.
- If the standby database that is assuming the primary role is a logical standby database, then the original primary becomes a logical standby database.
- You cannot switchover to a snapshot standby database.
- If the standby database that is assuming the primary role is a logical standby database and there are physical standby databases in the configuration, after the switchover, the physical standby databases will be disabled.

▲ Caution:

For this reason, Oracle generally recommends that you specify your physical standby database for switchover instead of your logical standby database. If you must switch over to your logical standby database, see [Reenabling Disabled Databases After a Role Change](#) to re-create your physical standby database.

If you intend to switch back to the original primary database relatively soon, you may allow the physical and snapshot standbys to remain disabled. Once you have completed the switchover back to the original primary, you may then reenble the physical and snapshot standby databases since they are still viable standbys for the original primary database.

- If the database is managed by Oracle Clusterware, the broker does not open any PDBs on any of the instances. Instead, the broker notifies the Clusterware agent after the switchover completes, and the Clusterware agent opens PDBs on particular instances based on the service configuration.

Command Examples

Example 10-14 Successful Switchover From Primary to Physical Standby

This example shows a successful switchover in which the physical standby database, `South_Sales`, transitions into the primary role.

```
DGMGRL> SWITCHOVER TO 'South_Sales';
2021-03-08T18:46:18.576-05:00
Performing switchover NOW, please wait...

2021-03-08T18:46:31.899-05:00
New primary database "South_Sales" is opening...

2021-03-08T18:46:31.901-05:00
Operation requires start up of instance "north_sales1" on database
"North_Sales"
Starting instance "north_sales1"...
Connected to an idle instance.
ORACLE instance started.
Connected to "north_sales1"
Database mounted.

Connected to "South_Sales"
2021-03-08T18:47:12.754-05:00
```

```
Switchover succeeded, new primary is "South_Sales"
```

```
2021-03-08T18:47:12.780-05:00
```

```
Switchover processing complete, broker ready.
```

Example 10-15 Unsuccessful Switchover Due to Use of O/S Authentication

If you connect to the database using operating system authentication, you can use any username and password to connect. However, DGMGRL may not be able to shut down and start up the primary and standby database automatically because it cannot remotely authenticate itself.

The following example shows a switchover that succeeded but returns an error because DGMGRL cannot shut down and start up the primary and standby databases.

```
DGMGRL> SWITCHOVER TO 'South_Sales';
Performing switchover NOW, please wait...
New primary database "South_Sales" is opening...
Operation requires shutdown of instance "north_sales1" on database "North_Sales"
Shutting down instance "north_sales1"...
ORA-01031: insufficient privileges
```

```
Warning: You are no longer connected to ORACLE.
```

```
Please complete the following steps to finish switchover:
```

```
    shut down instance "north_sales1" of database "North_Sales"
    start up and mount instance "north_sales1" of database "North_Sales"
```



Note:

For DGMGRL to restart instances automatically, you must connect to the database using the same credentials given in the last `CONNECT` command, even if the last `CONNECT` command was used to connect to another database.

You must manually issue the `SHUTDOWN` and `STARTUP` commands to restart the new primary and any standby instances that may have been shut down.

Example 10-16 Redirecting Client Connections to a Target Physical Standby Database

This example performs a successful switchover in which the physical standby database `South_Sales` transitions into the primary role. Connections to the old primary are automatically reconnected to the new primary database, `South_Sales`.

```
DGMGRL> SWITCHOVER TO South_Sales;
2021-03-08T18:42:38.906-05:00
Performing switchover NOW, please wait...
```

```
2021-03-08T18:42:39.704-05:00
```

```
Operation requires a connection to database "South_Sales"
Connecting ...
Connected to "South_Sales"
Connected as SYSDBG.
```

```
2021-03-08T18:42:39.908-05:00
Continuing with the switchover...

2021-03-08T18:42:50.022-05:00
New primary database "South_Sales" is opening...

2021-03-08T18:42:50.023-05:00
Operation requires start up of instance "north_sales2" on database
"North_Sales"
Starting instance "north_sales2"...
Connected to an idle instance.
ORACLE instance started.
Connected to "North_Sales"
Database mounted.

Connected to "South_Sales"
2021-03-08T18:43:31.457-05:00
Switchover succeeded, new primary is "South_Sales"

2021-03-08T18:43:31.486-05:00
Switchover processing complete, broker ready.
```

Example 10-17 Specifying a Zero Wait Time During Switchover

This example includes the `WAIT` option in the `SWITCHOVER` command and sets the wait time to zero seconds. Therefore, the broker does not wait for sessions to drain and proceeds with the switchover operation.

```
DGMGRL> SWITCHOVER TO 'South_Sales' WAIT 0;
2021-03-08T18:29:17.674-05:00
WAIT 0 does not wait for sessions to drain; proceeding with switchover...

2021-03-08T18:29:27.995-05:00
New primary database "South_Sales" is opening...

2021-03-08T18:29:27.995-05:00
Oracle Clusterware is restarting database "North_Sales" ...

Connected to "South_Sales"
2021-03-08T18:30:09.375-05:00
Switchover succeeded, new primary is "South_Sales"

2021-03-08T18:30:09.421-05:00
Switchover processing complete, broker ready.
```

Example 10-18 Using the WAIT Clause to Specify a Wait Time During Switchover

This example includes the `WAIT` option in the `SWITCHOVER` command and sets the wait time to 23 seconds. Therefore, the broker waits for 23 seconds for sessions to drain and then proceeds with the switchover.

```
DGMGRL> SWITCHOVER TO 'South_Sales' WAIT 23;
2021-03-08T18:26:29.412-05:00
Stopping services and waiting up to 23 seconds for sessions to drain...
```

```
2021-03-08T18:26:40.209-05:00
Done waiting for sessions to drain; proceeding with switchover now...

2021-03-08T18:26:54.411-05:00
New primary database "South_Sales" is opening...

2021-03-08T18:26:54.412-05:00
Oracle Clusterware is restarting database "North_Sales" ...

Connected to "South_Sales"
2021-03-08T18:27:39.045-05:00
Switchover succeeded, new primary is "South_Sales"

2021-03-08T18:27:39.084-05:00
Switchover processing complete, broker ready.
```

SWITCHOVER PLUGGABLE DATABASE

This command switches the role of a source PDB with its designated target PDB.

Prerequisites

- Connect to the target database.
- The specified target PDB must exist in the target database and must be configured as a DG PDB.

Syntax

```
SWITCHOVER TO PLUGGABLE DATABASE pdb-name AT database-name;
```

Parameters

Parameter	Description
<i>pdb-name</i>	Name of the target PDB to which source PDB operations must be switched. The name specified must be the same as that used when you ran the <code>ADD PLUGGABLE DATABASE</code> command to set up the source PDB.
<i>database-name</i>	Name of the target database that contains the target PDB.

Usage Notes

- The broker verifies that source PDB is open and that redo apply services are running on the target database.
- The broker verifies the named PDB is a DG PDB at the specified database, redo apply is running, and the source for this DGPDB is up and running.

Examples

Example 10-19 Switching Over to a Target PDB

This example switches operations of the source PDB with the target PDB named `dgpdb_sales` that is contained in the target database `cdb_newyork`.

```
DGMGRL> SWITCHOVER TO PLUGGABLE DATABASE dgpdb_sales AT newyork;
```

VALIDATE DATABASE

The `VALIDATE DATABASE` command performs a comprehensive set of database checks prior to a role change.

The checks use information available in various Oracle Data Guard views as well as the Automatic Diagnostic Repository.

Format

```
VALIDATE DATABASE [VERBOSE] database-name [ STRICT  
    { TEMP_FILES | FLASHBACK | LOG_FILES_CLEARED | LOG_FILE_CONFIGURATION |  
  APPLY_PROPERTY |  
    TRANSPORT_PROPERTY | ALL } [ <strict_option> ... <strict_option> ] ];
```

Command Parameters

database-name

The name of the database for which you want to display information. The `VERBOSE` keyword, if used, must come before the `database-name` or an error is returned.

Usage Notes

The `VALIDATE DATABASE` command shows a brief summary of the database, and reports any errors or warnings that were detected. `VALIDATE DATABASE VERBOSE` shows everything in the brief summary plus all items that were validated.

Command Examples

The examples in this section show what the `VALIDATE DATABASE` command output might look like in both the brief and verbose forms for primary and standby databases.

Example 10-20 VALIDATE DATABASE Output in Brief Format for a Primary

The following example shows brief output for a primary database:

```
DGMGRL> VALIDATE DATABASE South_Sales;
```

```
Database Role:      Primary database
```

```
Ready for Switchover: Yes
```

```
Managed by Clusterware:
  South_Sales: YES
```

Example 10-21 VALIDATE DATABASE Output in Brief Format for a Physical Standby

The following example shows brief output for a physical standby database:

```
DGMGRL> VALIDATE DATABASE North_Sales;

Database Role:      Physical standby database
Primary Database:  South_Sales

Ready for Switchover:  Yes
Ready for Failover:  Yes (Primary Running)

Managed by Clusterware:
  South_Sales : NO
  North_Sales: NO
  The static connect identifier allows for a connection to database
  "South_Sales".

Parameter Settings:
  Parameter                               South_Sales Value
North_Sales Value
  DB_BLOCK_CHECKING                       true                true
  DB_BLOCK_CHECKSUM                       true                true
  DB_LOST_WRITE_PROTECT                   NONE                NONE
```

Example 10-22 VALIDATE DATABASE Output in Verbose Format for a Primary

The following example shows verbose output for a primary database:

```
DGMGRL> VALIDATE DATABASE VERBOSE South_Sales;

Database Role:      Primary database

Ready for Switchover:  Yes

Flashback Database Status:
  Database      Status      Retention Target
  South_Sales   On          1440

Capacity Information:
  Database      Instances  Threads
  South_Sales   1          1

Managed by Clusterware:
  South_Sales: NO
  The static connect identifier allows for a connection to database
  "South_Sales".

Temporary Tablespace File Information:
  South_Sales TEMP Files: 1
```

```
Data file Online Move in Progress:
  South_Sales: No

Transport-Related Information:
  Transport On: Yes

Log Files Cleared:
  South_Sales Standby Redo Log Files: Cleared
```

Example 10-23 VALIDATE DATABASE Output in Verbose Format for a Physical Standby

The following command shows verbose output for a physical standby database:

```
DGMGRL> validate database verbose North_Sales

Database Role:      Physical standby database
Primary Database:  South_Sales

Ready for Switchover: Yes
Ready for Failover: Yes (Primary Running)

Flashback Database Status:
  Database      Status      Retention Target
  South_Sales   On          1440
  North_Sales   On          1440

Capacity Information:
  Database  Instances  Threads
  South_Sales    1          1
  North_Sales    1          1

Managed by Clusterware:
  South_Sales : NO
  North_Sales: NO
  The static connect identifier allows for a connection to database
  "South_Sales".

Temporary Tablespace File Information:
  South_Sales TEMP Files: 1
  North_Sales TEMP Files: 1

Data file Online Move in Progress:
  South_Sales: No
  North_Sales: No

Standby Apply-Related Information:
  Apply State:      Running
  Apply Lag:        0 seconds (computed 0 seconds ago)
  Apply Delay:      0 minutes

Transport-Related Information:
  Transport On: Yes
  Gap Status:      No Gap
```

Transport Lag: 0 seconds (computed 0 seconds ago)
Transport Status: Success

Log Files Cleared:

South_Sales Standby Redo Log Files: Cleared
North_Sales Online Redo Log Files: Cleared
North_Sales Standby Redo Log Files: Available

Current Log File Groups Configuration:

Thread #	Online Redo Log Groups (South_Sales)	Standby Redo Log Groups
(North_Sales)		
1	4	5

Future Log File Groups Configuration:

Thread #	Online Redo Log Groups (North_Sales)	Standby Redo Log Groups
(South_Sales)		
1	4	5

Current Configuration Log File Sizes:

Thread #	Smallest Online Redo Log File Size (South_Sales)	Smallest Standby Redo Log File Size
(North_Sales)		
1	25 MBytes	25 MBytes

Future Configuration Log File Sizes:

Thread #	Smallest Online Redo Log File Size (North_Sales)	Smallest Standby Redo Log File Size
(South_Sales)		
1	25 MBytes	25 MBytes

Apply-Related Property Settings:

Property	South_Sales Value
North_Sales Value	
DelayMins	0
ApplyParallel	AUTO
ApplyInstances	0

Transport-Related Property Settings:

Property	South_Sales Value
North_Sales Value	
LogShipping	ON
LogXptMode	ASYN
Dependency	<empty>
DelayMins	0
Binding	optional
MaxFailure	0
ReopenSecs	30
NetTimeout	300
RedoCompression	DISABLE

Parameter Settings:

Parameter	South_Sales	Value
North_Sales	Value	
DB_BLOCK_CHECKING	true	true
DB_BLOCK_CHECKSUM	true	true
DB_LOST_WRITE_PROTECT	NONE	NONE

VALIDATE DATABASE DATAFILE

The `VALIDATE DATABASE DATAFILE` command performs validation of data files across the primary database and standby databases.

The validation of data files detects lost writes at either database.

Format

```
VALIDATE DATABASE [database-name | ALL] DATAFILE [datafile-name | datafile-number | ALL] OUTPUT="output-file-name";
```

Command Parameters

database-name

The name of the database for which you want to display information. The `VERBOSE` keyword, if used, must come before the `database-name` or an error is returned.

If the database to be validated is either the primary or `ALL`, then the data files for all standby databases are compared with data files of the primary.

If the database to be validated is a standby database, then its data files are compared with the data files of the primary.

datafile-name | datafile-number

You can specify a data file to be compared by name (`datafile-name`) or by number (`datafile-number`).

The `datafile-name` is the name of a specific data file that you want validated.

The `datafile-number` is the file identification number of a data file (as shown in the `FILE#` column of the `V$DATAFILE` view).

output-file-name

A file generated on the server that you must check to determine if block comparison is completed and whether there were any lost writes. Output files are created in the `diagnostics trace` directory of the database being compared.

Usage Notes

- The `VALIDATE DATABASE` command shows a brief summary of the database, and reports any errors or warnings that were detected.
- When the `VALIDATE DATABASE` command is issued, it immediately returns a message that data file comparison has started on a database, but this does not mean that data file comparison completed or that there were no lost-writes between data files. You must check the output files that are generated to determine whether data file comparison was completed, or if there were lost writes.

Command Example

Example: Using `VALIDATE DATABASE DATAFILE` to Compare Data Files

The following command would compare the data files on the standby to those on the primary. Output would be sent to a file named `dbcomp1.out`.

```
DGMGRL> VALIDATE DATABASE boston DATAFILE ALL OUTPUT=dbcomp1.out;
Operation requires a connection to database "boston"
Connecting ...
Output files are created in /path/to/trace on host "standby-host"
```

The following shows sample output from the command:

```
Client is connected to database: boston. Role: physical standby.

*****
Remote database chicago.
remote db role: primary database

Slave Id 0
Summary:
Different data block pairs: 66617

Details:
*****
ID: Block Type Id
TOTAL: Total number of blocks found
DIFFV: Number of block pairs with different version
LWLOC: Lost Writes at Local
LWRMT: Lost Writes at Remote
SAMEV: Number of block pairs with same version
SAMEV&C: Number of block pairs with same version and checksum
DIFFPAIR: Number of block pairs with same version but different
contents
ENCERR: Undecided blocks related to encryption/decryption error.
e.g. Wallet is not open.
SKIPPED: Skipped blocks due to data corruption, direct load, etc

ID TOTAL DIFFV LWLOC LWRMT SAMEV SAMEV&C DIFFPAIR ENCERR
SKIPPED
02 0067698 0001032 0000000 0000000 0066666 0000049 0066617 0000000
0000000
29 0000001 0000001 0000000 0000000 0000000 0000000 0000000 0000000
0000000
30 0000125 0000001 0000000 0000000 0000124 0000124 0000000 0000000
0000000
38 0000014 0000014 0000000 0000000 0000000 0000000 0000000 0000000
0000000
```

VALIDATE DATABASE SPFILE

The `VALIDATE DATABASE SPFILE` command performs a comparison of server parameter file (SPFILE) entries between the primary database and a specified standby database.

The validation of the server parameter file detects parameter value discrepancies between the primary and the specified standby database so that they can be rectified before a role change, thus ensuring that after a role change the databases perform at the same level they did prior to the role change. Additionally, using this command frees you from having to restart a database to correct improperly set parameters.

Format

```
VALIDATE DATABASE {database-name} SPFILE;
```

Command Parameters

database-name

The name of the standby database for which you want to display information. The `VERBOSE` keyword, if used, must come before the database-name or an error is returned.

If the database to be validated is the primary database, then a message is returned saying the command cannot be issued on a primary database.

If the database to be validated is a standby database, then the server parameter file values on the primary database are compared with the server parameter file values on the standby database.

Usage Notes

- The `VALIDATE DATABASE SPFILE` command reports `No parameter differences found` if there are no differences and a list of the parameters with their differing values on the primary and the specified standby databases.
- When the `VALIDATE DATABASE SPFILE` command is issued, it makes a connection to the primary database and the specified standby database based on the respective values of the `DGConnectIdentifier` property. The command fails if a connection attempt cannot complete successfully.

Command Example

Example: Using VALIDATE DATABASE SPFILE to Compare Server Parameter File Values

The following is sample output from the `VALIDATE DATABASE SPFILE` command when there are no differences between the server parameter file values on the specified standby database and the primary database:

```
DGMGRL> VALIDATE DATABASE chicago SPFILE;
Connecting to "boston".

Connecting to "chicago".

No parameter differences found.
```

The following is sample output from the `VALIDATE DATABASE SPFILE` command when there are differences (different values, or specified on one and not on the other) between the server parameter file values on the specified standby database and the primary database:

```
DGMGRL> VALIDATE DATABASE chicago SPFILE;
Connecting to "boston".

Connecting to "chicago".
```

Parameter settings with different values:

```
aq_tm_processes:
boston (PRIMARY) : 8
chicago          : 9

commit_point_strength:
boston (PRIMARY) : NOT SPECIFIED
chicago          : 255

sec_max_failed_login_attempts:
boston (PRIMARY) : 2
chicago          : NOT SPECIFIED

use_large_pages:
boston (PRIMARY) : TRUE
chicago          : NOT SPECIFIED
DGMGRL>
```

VALIDATE DGConnectIdentifier

The `VALIDATE DGConnectIdentifier` command enables users to check to see whether a connection string is valid for the `DGConnectIdentifier` property.

The connect identifier for each connectivity check is generated based on the `DGConnectIdentifier` property of the associated database.

Format

```
VALIDATE DGConnectIdentifier <connect_string>;
```

Command Parameters

<connect_string>

The network connect string.

Usage Notes

The DGMGRL new command "VALIDATE DGConnectIdentifier" enables users to check to see whether a connection string is valid for the `DGConnectIdentifier` property or not. The command checks if it is able to be translated to something useful and makes a connection using it on all configuration members. This can also be used prior to adding a member to the configuration. If no config exists, the command checks a connection string at the database DGMGRL is connected to. If a config exists, all members use the specified connection string to see that the connection works.

This command performs the following for each instance of all members:

- Print a translated network of the connection string at the instance
- Print environment variables related to network configuration at the instance
- Make a new connection using the translated network address at the instance
- If a connection test succeeds, the instance name and `db_unique_name` of the connected database will be printed

Command Examples

Example 1: When a configuration exists:

```
DGMGRL> validate DGConnectIdentifier 'tkdg2_tns' ;
Connecting to instance "tkdg1_sid" on database "tkdg1_db_uniq" ...
  Translated Network Address:
  (DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=tcp) (HOST=<nodename>)
  (PORT=<num>)))
  (CONNECT_DATA=(SERVICE_NAME=tkdg2_db_uniq.regress.rdbms.dev.us.oracle.com)
  (SERVER=DEDICATED)))
  Environment Variables:
    TNS_ADMIN: /opt/oracle/admin
    ORACLE_HOME: /opt/oracle
    GRID_HOME:
  Connection Test Result: Connected to instance 'tkdg2_sid' of database
  'tkdg2_db_uniq'.
  DGConnectIdentifier 'tkdg2_tns' is valid at instance 'tkdg1_sid' of
  database'tkdg1_db_uniq'

Connecting to instance "tkdg2_sid" on database "tkdg2_db_uniq" ...
  Translated Network Address:
  (DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=tcp) (HOST=<nodename>)
  (PORT=<num>)))
  (CONNECT_DATA=(SERVICE_NAME=tkdg2_db_uniq.regress.rdbms.dev.us.oracle.com)
  (SERVER=DEDICATED)))
  Environment Variables:
    TNS_ADMIN: /opt/oracle/admin
    ORACLE_HOME: /opt/oracle
    GRID_HOME:
  Connection Test Result: Connected to instance 'tkdg2_sid' of database
  'tkdg2_db_uniq'.
  DGConnectIdentifier 'tkdg2_tns' is valid at instance 'tkdg2_sid' of
  database'tkdg2_db_uniq'
```

Example 2: When no configuration exists:

```
DGMGRL> validate DGConnectIdentifier 'tkdg2_tns' ;
Validating DGConnectIdentifier at instance 'tkdg1_sid' of database
'tkdg1_db_uniq'
Translated Network Address:
  (DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=tcp) (HOST=<nodename>)
  (PORT=<num>)))
  (CONNECT_DATA=(SERVICE_NAME=tkdg2_db_uniq.regress.rdbms.dev.us.oracle.com)
  (SERVER=DEDICATED)))
  Environment Variables:
    TNS_ADMIN: /opt/oracle/admin
    ORACLE_HOME: /opt/oracle
    GRID_HOME:
  Connection Test Result: Connected to instance 'tkdg2_sid' of database
  'tkdg2_db_uniq'.
```

```
DGConnectIdentifier 'tkdg2_tns' is valid at instance 'tkdg1_sid' of
database'tkdg1_db_uniq'
```

Example 3: With an invalid connection string:

```
DGMGRL> validate DGConnectIdentifier 'tkdg2_tns' ;
Validating DGConnectIdentifier at instance 'tkdg1_sid' of database
'tkdg1_db_uniq'
Translated Network Address:
(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=tcp) (HOST=<nodename>)
(PORT=<num>)))
(CONNECT_DATA=(SERVICE_NAME=tkdg2_db_uniq.regress.rdbms.dev.us.oracle.c
om) (SERVER=DEDICATED)))
Environment Variables:
TNS_ADMIN: /opt/oracle/admin
ORACLE_HOME: /opt/oracle
GRID_HOME:
Connection Test Result: Connected to instance 'tkdg2_sid' of
database 'tkdg2_db_uniq'.
DGConnectIdentifier 'tkdg2_tns' is valid at instance 'tkdg1_sid' of
database'tkdg1_db_uniq'
```

Example 4: When a connection test fails:

```
DGMGRL> validate DGConnectIdentifier 'tkdg2_tns' ;
Validating DGConnectIdentifier at instance 'tkdg1_sid' of database
'tkdg1_db_uniq'
Translated Network Address:
(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=tcp) (HOST=<nodename>)
(PORT=<num>)))
(CONNECT_DATA=(SERVICE_NAME=tkdg2_db_uniq.regress.rdbms.dev.us.oracle.c
om) (SERVER=DEDICATED)))
Environment Variables:
TNS_ADMIN: t_work
ORACLE_HOME: ORACLE_HOME
GRID_HOME:
Connection Test Result: Failed - ORA-12514: TNS:listener does not
currently know of service requested in connect descriptor
```

VALIDATE FAR_SYNC

The `VALIDATE FAR_SYNC` command performs a comprehensive set of checks for a far sync instance.

The checks use information available in various Oracle Data Guard views.

Format

```
VALIDATE FAR_SYNC [VERBOSE] far_sync_instance_name [WHEN PRIMARY
IS database-name];
```

Command Parameters

far_sync_instance_name

The name of the far sync instance for which you want to display information. The `VERBOSE` keyword, if used, must come before the `far_sync_instance_name` or an error is returned.

database-name

The validation of the far sync instance is performed based on the specified database being the primary database.

Usage Notes

The `VALIDATE FAR_SYNC` command shows a brief summary of the far sync instance and reports any errors or warnings that were detected. The `VALIDATE FAR_SYNC VERBOSE` command shows everything in the brief summary plus redo transport-related information.

Command Examples

The examples in this section show what the `VALIDATE FAR_SYNC` command output might look like in various scenarios.

Example 1: Brief VALIDATE FAR_SYNC Output

The following example shows brief output for a far sync instance:

```
DGMGRL> VALIDATE FAR_SYNC FS;

Member Role:          Far Sync Instance
When Primary Is:      North_Sales

Active Redo Source:   North_Sales
Redo Destinations:
                    South_Sales

Thread #  Online Redo Log Groups  Standby Redo Log Groups  Status
          North_Sales          FS
1         4                    5                    Sufficient SRLs
```

Example 2: Verbose VALIDATE FAR_SYNC Output

The following example shows verbose output for a far sync instance:

```
DGMGRL> VALIDATE FAR_SYNC VERBOSE FS;

Member Role:          Far Sync Instance
When Primary Is:      North_Sales

Active Redo Source:   North_Sales
Redo Destinations:
                    South_Sales

Thread #  Online Redo Log Groups  Standby Redo Log Groups  Status
          North_Sales          FS
1         4                    5                    Sufficient SRLs

Transport-Related Information:
Transport On:         Yes
Gap Status:          No Gap
Transport Lag:       0 seconds (computed 0 seconds ago)
Transport Status:    Success
```

Example 3: VALIDATE FAR_SYNC Output When South_Sales Is the Primary

The following example shows the output when the `South_Sales` database is specified as the primary database:

```
DGMGRL> VALIDATE FAR_SYNC FS WHEN PRIMARY IS 'South_Sales';

Member Role:          Far Sync Instance
When Primary Is:     South_Sales
Redo Destinations:
                    North_Sales

Thread #  Online Redo Log Groups  Standby Redo Log Groups  Status
         South_Sales             FS
1         4                       5                          Sufficient SRLs
```

VALIDATE FAST_START FAILOVER

The `VALIDATE FAST_START FAILOVER` command enables you to validate a fast-start failover configuration. It identifies misconfiguration, either while setting up or initiating fast-start failover.

Format

```
VALIDATE FAST_START FAILOVER;
```

Command Parameters

None.

Usage Notes

- This command validates the fast-start failover configuration and reports the following information:
 - Incorrectly set up fast-start failover parameters
For example, the fast-start failover threshold is not set appropriately.
 - Issues that prevent the enabling or initiating of fast-start failover
This includes issues that prevent the usage of fast-start failover even when the conditions required for fast-start failover are met (for example, fast-start failover is enabled in Observe-Only mode).
 - Issues that affect actions taken after fast-start failover is initiated
 - Issues that could impact the stability of the broker configuration
 - Issues with fast-start failover callout configuration scripts
Displays if the syntax of the fast-start failover configuration file `fsfocallout.ora` is correct and if the pre-callout and post-callout scripts are accessible.

Command Examples

Example 10-24 Validating a Broker Configuration

The following example validates the broker configuration that is in maximum performance mode and identifies issues that prevent fast-start failover from being enabled for the configuration.

```
DGMGRL> VALIDATE FAST_START FAILOVER;
Fast-Start Failover: Enabled in Potential Data Loss Mode
Protection Mode:      MaxPerformance
Primary:              North_Sales
Active Target:       South_Sales

Fast-Start Failover Not Possible:
Fast-Start Failover observer not started

Post Fast-Start Failover Issues:
Flashback database disabled for database 'dgv1'

Other issues:
FastStartFailoverThreshold may be too low for RAC databases.

Fast-start failover callout configuration file "fsfocallout.ora" has the
following issues:
Invalid lines
foo=foo
The specified file "./precallout" contains a path.
```

Example 10-25 Validating an Unsynchronized Fast-start Failover Configuration

This example validates a fast-start failover configuration that is enabled in regular mode. The callout scripts exist and are syntactically accurate. The only problem is that the configuration is unsynchronized.

```
DGMGRL> VALIDATE FAST_START FAILOVER;
Fast-Start Failover: Enabled in Observe-Only mode
Protection Mode:      MaxPerformance
Primary:              North_Sales
Active Target:       South_Sales

Fast-Start Failover not possible:
Fast-start failover configuration is unsynchronized
```

VALIDATE NETWORK CONFIGURATION

The `VALIDATE NETWORK CONFIGURATION` command performs network connectivity checks between members of a configuration.

The connect identifier for each connectivity check is generated based on the `DGConnectIdentifier` property of the associated database.

Format

```
VALIDATE NETWORK CONFIGURATION FOR { ALL | member name };
```

Command Parameters**member name**

The name of the configuration member to validate.

Usage Notes

- This command also performs a check for the static connect identifier.

Command Examples**Example 1: Validating Network Configuration for a Specific Database**

```
DGMGRL> VALIDATE NETWORK CONFIGURATION FOR North_Sales;

Connecting to instance "north_sales1" on database "North_Sales" ...

Checking connectivity from instance "north_sales1" on database
"North_Sales" to instance "south_sales1" on database "South_Sales"...

Succeeded.

Connecting to instance "north_sales6" on database "North_Sales" ...

Checking connectivity from instance "north_sales6" on database
"North_Sales" to instance "south_sales1" on database "South_Sales"...

Succeeded.

Connecting to instance "south_sales1" on database "South_Sales" ...

Checking connectivity from instance "south_sales1" on database
"South_Sales" to instance "north_sales1" on database "North_Sales"...

Succeeded.

Checking connectivity from instance "south_sales1" on database
"South_Sales" to instance "north_sales6" on database "North_Sales"...

Succeeded.

Oracle Clusterware on database "North_Sales" is available for database
restart.
```

Example 2: Validating Network Configuration For All Members

```
DGMGRL> VALIDATE NETWORK CONFIGURATION FOR ALL;

Connecting to instance "north_sales1" on database "North_Sales" ...
```

```
Checking connectivity from instance "north_sales1" on database "North_Sales"
to instance "south_sales1" on database "South_Sales"...
```

Succeeded.

```
Connecting to instance "north_sales6" on database "North_Sales" ...
```

```
Checking connectivity from instance "north_sales6" on database "North_Sales"
to instance "south_sales1" on database "South_Sales"...
```

Succeeded.

```
Connecting to instance "south_sales1" on database "South_Sales" ...
```

```
Checking connectivity from instance "south_sales1" on database "South_Sales"
to instance "north_sales1" on database "North_Sales"...
```

Succeeded.

```
Checking connectivity from instance "south_sales1" on database "South_Sales"
to instance "north_sales6" on database "North_Sales"...
```

Succeeded.

```
Oracle Clusterware on database "North_Sales" is available for database
restart.
```

```
Oracle Clusterware is not configured on database "South_Sales".
Connecting to database "South_Sales" using static connect identifier
"(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=tcp)(HOST=<nodename>)
(PORT=.*))))(CONNECT_DATA=(SERVICE_NAME=South_Sales_DGMGR.example.com)
(INSTANCE_NAME=south_sales1)(SERVER=DEDICATED)(STATIC_SERVICE=TRUE)))" ...
Succeeded.
```

```
The static connect identifier allows for a connection to database
"South_Sales".
```

VALIDATE STATIC CONNECT IDENTIFIER

The `VALIDATE STATIC CONNECT IDENTIFIER` command validates the static connect identifier of a database.

To perform this validation, the broker makes a new connection to the database using a static connect identifier based on the `StaticConnectIdentifier` property of the database. A new attribute, `STATIC_SERVICE=TRUE` is added to the connect identifier to ensure that a connection to the database is established using only a static service, not a dynamic service.

Format

```
VALIDATE STATIC CONNECT IDENTIFIER FOR { ALL | database name };
```

Command Parameters

database name

The name of a specific database to validate.

Usage Notes

- None

Command Examples

Example 1: Validation of Static Connect Identifier For a Database on Which Oracle Clusterware Is Configured

```
DGMGRL> VALIDATE STATIC CONNECT IDENTIFIER FOR North_Sales;
```

Oracle Clusterware on database "North_Sales" is available for database restart.

Example 2: Validation of Static Connect Identifier For a Database on Which Oracle Clusterware Is Not Configured

```
DGMGRL> VALIDATE STATIC CONNECT IDENTIFIER FOR South_Sales;
```

```
Oracle Clusterware is not configured on database "South_Sales".  
Connecting to database "South_Sales" using static connect identifier  
"(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=tcp) (HOST=<nodename>  
(PORT=.*))))  
(CONNECT_DATA=(SERVICE_NAME=South_Sales_DGMGRL.example.com)  
(INSTANCE_NAME=tkdg2_sid) (SERVER=DEDICATED) (STATIC_SERVICE=TRUE)))" ...
```

Succeeded.

The static connect identifier allows for a connection to database "South_Sales".

Example1: Validation of Static Connect Identifier For all Databases

```
DGMGRL> VALIDATE STATIC CONNECT IDENTIFIER FOR all;
```

Oracle Clusterware on database "North_Sales" is available for database restart.

```
Oracle Clusterware is not configured on database "South_Sales".  
Connecting to database "South_Sales" using static connect identifier  
"(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=tcp) (HOST=<nodename>  
(PORT=.*))))  
(CONNECT_DATA=(SERVICE_NAME=South_Sales_DGMGRL.example.com)  
(INSTANCE_NAME=tkdg2_sid) (SERVER=DEDICATED) (STATIC_SERVICE=TRUE)))" ...
```

Succeeded.

The static connect identifier allows for a connection to database "South_Sales".

Oracle Data Guard Broker Properties

Use Oracle Data Guard broker properties to manage instances or the entire configuration.

Broker configuration and database properties help you to view and control the behavior of entire broker configurations, individual configuration members, redo transport services, and log apply services.

- [Configuration Properties](#)
- [Monitorable \(Read-Only\) Properties](#)
- [Configurable Properties](#)

Properties have either configuration-wide scope, database-wide scope, far sync instance-wide scope, or instance-specific scope. Configuration-wide properties control the behavior of the broker on all members of the configuration. The values of such properties apply uniformly across all members of the configuration.

Database-wide properties allow you to view or control the behavior of a specific database. If the database (primary or standby) is an Oracle RAC database consisting of multiple instances, then the value of such a property applies uniformly across all instances of that database.

Far sync instance-wide properties allow you to view or control the behavior of a far sync instance.

Instance-specific properties allow you to view or control the behavior of an individual database instance. Such a property exists for all instances of an Oracle RAC database, but its value can differ from one specific instance to another.

Note:

You can use the Oracle Data Guard broker command-line interface (DGMGRL) to view or modify broker properties.

Oracle Enterprise Manager Cloud Control (Cloud Control) explicitly presents some broker properties on the Edit Properties page. Information from other properties can be implicitly incorporated into other Web pages displayed by Cloud Control. See the individual description of each property for information about how Cloud Control presents that property.

Configuration Properties

Configuration properties control the behavior of the broker configuration.

You can view and dynamically update the values of these properties using either DGMGRL or Cloud Control. However, some properties can only be updated through DGMGRL.

A configuration property has configuration-wide scope; meaning that the value you set for the property applies uniformly to each member in the configuration.

Configuration properties are set using the `EDIT CONFIGURATION SET PROPERTY` command, as shown in the following examples.

Example 1

This example sets the `FastStartFailoverThreshold` configuration property to 90 seconds.

```
DGMGRL> EDIT CONFIGURATION SET PROPERTY FastStartFailoverThreshold=90;
```

Example 2

This example sets the `ExternalDestination1` configuration property to point to a redo transport destination with a service name of `Sales` and a `DB_UNIQUE_NAME` of `chicago`.

```
EDIT CONFIGURATION SET PROPERTY ExternalDestination1='SERVICE=Sales  
DB_UNIQUE_NAME=chicago';
```

Note:

In general, quotation marks are not necessary around property values unless the case you specify needs to be preserved. By default, DGMGRL lowercases any string you supply, unless it is placed within quotation marks.

Quotation marks are required for any property values that include spaces or punctuation characters (for example, `RedoRoutes`).

The following sections describe the configuration properties:

- [BystandersFollowRoleChange](#)
- [CommunicationTimeout](#)
- [ConfigurationWideServiceName](#)
- [ExternalDestination1](#)
- [ExternalDestination2](#)
- [FastStartFailoverAutoReinstate](#)
- [FastStartFailoverLagLimit](#)
- [FastStartFailoverPmyShutdown](#)
- [FastStartFailoverThreshold](#)
- [ObserverOverride](#)
- [ObserverPingInterval](#)
- [ObserverReconnect](#)
- [ObserverRetry](#)
- [OperationTimeout](#)
- [PreferredObserverHosts](#)

- [PrimaryLostWriteAction](#)
- [TraceLevel](#)

BystandersFollowRoleChange

The `BystandersFollowRoleChange` configuration property establishes whether bystander standby databases are evaluated *during* failover (value = `ALL`) or *after* failover (value = `NONE`).

Descriptions of the `ALL` and `NONE` options are as follows:

- `ALL` - During the failover process, the broker determines whether the bystander standby databases are ahead of or behind the failover target standby (that is, the standby that will be the new primary).

If the bystander standbys are ahead, they will be disabled with a status of `ORA-16661` as part of the failover operation. The bystander standbys must be reinstated after failover completes. The broker reinstates a standby through a flashback to the SCN at which the target standby became a primary database, and sets up the redo transport configuration from the new primary to the standby.

If the bystander standbys are behind, then the broker simply sets up the redo transport configuration from the soon-to-be-new-primary to these standbys and completes the failover process.

- `NONE` - During the failover process, the broker does NOT evaluate the status of the bystander standbys as part of the failover operation. They are marked as disabled with a status of `ORA-16661` so that they can be evaluated later. The broker simply completes the failover to produce a new primary database as soon as possible.

After the failover is completed, you can reinstate the bystander standbys. During reinstatement of a bystander, the broker determines whether the bystander is ahead of, or behind, the new primary. If the bystander is ahead of the new primary, then the broker automatically flashes back the standby to the SCN at which the target standby became a primary database and sets up redo transport from the new primary to the standby. (Even if a flashback is not required, the broker sets up the redo transport configuration from the new primary to these standbys.)

The `NONE` option decreases the processing time for failover, but disables broker management of all bystander databases in the configuration. If fast-start failover is enabled, then the observer automatically reinstates the standby databases after failover has completed. Otherwise, you will have to manually reinstate the standby databases after failover has completed.

Category	Description
Datatype	String
Valid value	<code>ALL</code> or <code>NONE</code>
Broker default	<code>ALL</code>
Imported?	No
Parameter class	Not applicable
Role	Primary and standby
Standby type	Not applicable
Corresponds to	Not applicable

Category	Description
Scope	Broker configuration. This property will be consumed by broker on the database that is the target of a complete failover.
Cloud Control name	Not applicable

CommunicationTimeout

The `CommunicationTimeout` configuration property allows you to decide how many seconds the broker should wait before timing out its network communication between members in the configuration.

A value of zero indicates that a network communication should never be timed out.

Category	Description
Datatype	Integer
Valid values	≥ 0
Broker default	180 seconds
Imported?	No
Parameter class	Not applicable
Role	Primary and standby
Standby type	Not applicable
Corresponds to	Not applicable
Scope	Broker configuration. This property will be consumed by broker on all databases in the configuration.
Cloud Control name	Not applicable

ConfigurationSimpleName

The `ConfigurationSimpleName` configuration property is used to specify the directory that contains the observer runtime data file (`fsfo.dat`), observer log files, and callout configuration files that correspond to a specific broker configuration.

For databases that use versions lower than Oracle Database Release 21c, `ConfigurationSimpleName` cannot be explicitly set. This value is determined in real time, when the broker connects to the database, by combining the configuration name and the database's `DB_UNIQUE_NAME` (after removing any invalid characters).

The initial value of `ConfigurationSimpleName` is set using the following rules (in the order listed):

- Broker configuration name, if the name is less than or equal to 30 characters and contains valid characters.
- First 30 characters of the broker configuration name, if the name is more than 30 characters and the first 30 characters contain valid values.
- `DB_UNIQUE_NAME`

This value is configurable and you can modify the initial value that is set.

Category	Description
Datatype	String
Valid values	<ul style="list-style-type: none"> The characters are limited to the following: [a...z], [A...Z], [0...9], _, #, and \$. The first character must be an alphanumeric character. There is a 30-character maximum. The name is case-sensitive.
Broker default	<p>If the configuration name contains 30 characters or less, and contains only the characters a-z or A-Z, numerals 0-9, or special characters _, #, \$, the default value is the first 30 characters of configuration name.</p> <p>If the configuration name is longer than 30 characters, and its first 30 characters contain only the characters a-z or A-Z, numerals 0-9, or special characters _, #, or \$, the default value is the first 30 characters of the configuration name.</p> <p>If neither of the above conditions is satisfied, the default value is the <code>DB_UNIQUE_NAME</code> of the primary database</p>
Imported?	No
Parameter class	Not applicable
Role	Primary and standby
Standby type	Not applicable
Corresponds to	Not applicable
Scope	Configuration
Cloud Control name	Yes

ConfigurationWideServiceName

The `ConfigurationWideServiceName` configuration property is used to change the name of the configuration-wide service.

The broker publishes a service on each member of a configuration with a unified service name. The default service name of this configuration-wide service is `primarydbname_CFG`, where the primary database name is appended with a suffix of `_CFG`. The service name does not change after a role transition.

Category	Description
Datatype	String
Valid values	<ul style="list-style-type: none"> The characters are limited to the following: [a...z], [A...Z], [0...9], and _ The first character must be an alphanumeric character. There is a 30-character maximum.
Broker default	<code>primarydbname_CFG</code>
Imported?	No
Parameter class	Not applicable
Role	Primary and standby
Standby type	Not applicable
Corresponds to	Not applicable
Scope	Configuration
Cloud Control name	Not applicable

ExternalDestination1

The `ExternalDestination1` configuration property is used to specify a redo transport destination that can receive redo data from the current primary database.

To set up transport of redo data to the specified destination, the broker uses the values specified for this parameter to define a `LOG_ARCHIVE_DEST_n` initialization parameter on the primary database. The broker also monitors the health of the transport to the specified destination.

After a role change, the broker automatically sets up a `LOG_ARCHIVE_DEST_n` initialization parameter on the new primary database to ship redo data to the specified destination.



Note:

The `ExternalDestination1` property requires that the database unique name (`DB_UNIQUE_NAME`) be specified.

Category	Description
Datatype	String
Valid values	Any <code>LOG_ARCHIVE_DEST_n</code> attributes, except the following: <ul style="list-style-type: none"> • ALTERNATE • GROUP • PRIORITY • VALID_FOR See <i>Oracle Data Guard Concepts and Administration</i> for more information about the <code>LOG_ARCHIVE_DEST_n</code> parameter.
Broker default	Empty string
Imported?	No
Parameter class	Not applicable
Role	Primary
Standby type	Not applicable
Corresponds to	Not applicable
Scope	Configuration
Cloud Control name	Not applicable



Note:

The `Externaldestination1` configuration property is available only in Oracle Database 11g Release 2 (11.2.0.4) and in Oracle Database 12c Release 1 (12.1.0.2) and later.

ExternalDestination2

The `ExternalDestination2` configuration property is used to specify a redo transport destination that can receive redo data from the current primary database.

To set up transport of redo data to the specified destination, the broker uses the values specified for this parameter to define a `LOG_ARCHIVE_DEST_n` initialization parameter on the primary database. The broker also monitors the health of the transport to the specified destination.

After a role change, the broker automatically sets up a `LOG_ARCHIVE_DEST_n` initialization parameter on the new primary database to ship redo data to the specified destination.



Note:

The `ExternalDestination2` property requires that the database unique name (`DB_UNIQUE_NAME`) be specified.

Category	Description
Datatype	String
Valid values	Any <code>LOG_ARCHIVE_DEST_n</code> attributes, except the following: <ul style="list-style-type: none"> • ALTERNATE • GROUP • PRIORITY • VALID_FOR See <i>Oracle Data Guard Concepts and Administration</i> for more information about the <code>LOG_ARCHIVE_DEST_n</code> parameter.
Broker default	Empty string
Imported?	No
Parameter class	Not applicable
Role	Primary
Standby type	Not applicable
Corresponds to	Not applicable
Scope	Configuration
Cloud Control name	Not applicable



Note:

The `Externaldestination2` configuration property is available only in Oracle Database 11g Release 2 (11.2.0.4) and in Oracle Database 12c Release 1 (12.1.0.2) and later.

FastStartFailoverAutoReinstate

The `FastStartFailoverAutoReinstate` configuration property causes the former primary database to be automatically reinstated if a fast-start failover was initiated because the primary database was either isolated or had crashed.

To prevent automatic reinstatement of the former primary database in these cases, set this configuration property to `FALSE`.

The broker never automatically reinstates the former primary database if a fast-start failover was initiated because a user configuration condition was detected or was requested by an application calling the `DBMS_DG.INITIATE_FS_FAILOVER` function.

Category	Description
Datatype	Boolean
Valid value	TRUE or FALSE
Broker default	TRUE
Imported?	No
Parameter class	Not applicable
Role	Primary and standby
Standby type	Not applicable
Corresponds to	Not applicable
Scope	Broker configuration. This property will be consumed by the observer after fast-start failover has been enabled.
Cloud Control name	Automatically Reinstate Primary

FastStartFailoverLagLimit

The `FastStartFailoverLagLimit` configuration property establishes an acceptable limit, in seconds, that the standby is allowed to fall behind the primary in terms of redo applied.

If the limit is reached, then a fast-start failover is not allowed. The lowest possible value is 5 seconds.

This property is used when fast-start failover is enabled and the configuration is operating in maximum performance mode.

Category	Description
Datatype	Integer
Valid value	Integral number of seconds. The value depends on the protection mode. Maximum performance mode: Must be greater than, or equal to, 5. Maximum availability mode: Must be equal to zero. The <code>FastStartFailoverLagLimit</code> property is not used in maximum protection mode.
Broker default	30 seconds

Category	Description
Imported?	No
Parameter class	Not applicable
Role	Primary and standby
Standby type	Not applicable
Corresponds to	Not applicable
Scope	Broker configuration. This property will be consumed by the primary database after fast-start failover has been enabled.
Cloud Control name	Lag Limit

FastStartFailoverPmyShutdown

The `FastStartFailoverPmyShutdown` configuration property causes the primary database to shut down under certain conditions.

The primary database shuts down if fast-start failover is enabled and `V$DATABASE.FS_FAILOVER_STATUS` indicates the primary has been `STALLED` for longer than `FastStartFailoverThreshold` seconds. In such a situation, it is likely that the primary has been isolated and a fast-start failover has already occurred. A value of `TRUE` helps to ensure that an isolated primary database cannot satisfy user queries.

Setting this property to `FALSE` will not prevent the primary database from shutting down if a fast-start failover occurred because a user configuration condition was detected or was requested by an application by calling the `DBMS_DG.INITIATE_FS_FAILOVER` function.

Category	Description
Datatype	Boolean
Valid value	TRUE or FALSE
Broker default	TRUE
Imported?	No
Parameter class	Not applicable
Role	Primary and standby
Standby type	Not applicable
Corresponds to	Not applicable
Scope	Broker configuration. This property will be consumed by the primary database after fast-start failover has been enabled.
Cloud Control name	Automatically Shutdown Primary

FastStartFailoverThreshold

The `FastStartFailoverThreshold` configuration property defines the number of seconds the master observer attempts to reconnect to the primary database before initiating a fast-start failover.

(If there is only one observer, then by default it is considered the master. The time interval starts when the observer first loses connection with the primary database. If the observer is

unable to regain a connection to the primary database within the specified time, then the observer initiates a fast-start failover to the target standby database. See Task 4 in [Enabling Fast-Start Failover](#) for more information about setting this property.

The observer ignores the threshold completely if a configurable fast-start failover condition is detected or an application has requested that fast-start failover be initiated.

For help in determining an appropriate value for this property, you can use the information provided in the `V$FS_OBSERVER_HISTOGRAM` view. This view displays statistics that are based on the frequency of successful pings between the observer and primary database for different time intervals. (This view is available in Oracle Database 12c Release 1 (12.1.0.2) and later. It is also available in Oracle Database 11g Release 2 (11.2.0.4), but not in Oracle Database 12c Release 1 (12.1.0.1).) See *Oracle Database Reference* for a description of the `V$FS_OBSERVER_HISTOGRAM` view.

Category	Description
Datatype	Integer
Valid value	Integral number of seconds. Must be greater than, or equal to, 6.
Broker default	30 seconds
Imported?	No
Parameter class	Not applicable
Role	Target standby database that is about to fail over to the primary role
Standby type	Not applicable
Corresponds to	Not applicable
Scope	Broker configuration. This property will be consumed by the observer after fast-start failover has been enabled.
Cloud Control name	Cloud Control presents this as "Failover Threshold" on the Oracle Data Guard Overview page.

ObserverOverride

The `ObserverOverride` configuration property, when set to `TRUE`, allows an automatic failover to occur when the observer has lost connectivity to the primary.

This is the behavior even if the standby has a healthy connection to the primary.

Category	Description
Datatype	Boolean
Valid values	<code>TRUE</code> or <code>FALSE</code>
Broker default	<code>FALSE</code>
Imported?	No
Parameter class	Not applicable
Role	Primary and standby
Standby type	Not applicable
Corresponds to	Not applicable
Scope	Broker configuration. This property will be consumed by the observer after fast-start failover has been enabled.

Category	Description
Cloud Control name	Not applicable

ObserverPingInterval

The `ObserverPingInterval` configuration property specifies how frequently the observer must ping the primary database.

This property is measured in milliseconds. The minimum value is 100 milliseconds. To achieve lower detection times for primary database failures, you must set the `ObserverPingInterval` and `ObserverPingRetry` properties before enabling fast-start failover.

The `FastStartFailoverThreshold` property is ignored when the `ObserverPingRetry` and `ObserverPingInterval` properties are set.

Category	Description
Datatype	Integer
Valid values	>= 100
Broker default	100 milliseconds
Imported?	No
Parameter class	Not applicable
Role	Primary
Standby type	Not applicable
Corresponds to	Not applicable
Scope	Broker configuration. This property will be consumed by the observer after fast-start failover has been enabled.
Cloud Control name	Not applicable

ObserverPingRetry

The `ObserverPingRetry` property specifies the number of times that the observer retries a failed ping before it initiates a failover to the target standby database.

A failed ping is a ping to the primary database that failed or took longer than the time specified by the `ObserverPingInterval` property. You must set both the `ObserverPingRetry` and `ObserverPingInterval` properties to achieve lower detection times for primary database failures. The minimum value of `ObserverRetryCount` is 10. Therefore, the detection time can be reduced to nearly 1 second.

The `FastStartFailoverThreshold` property is ignored when the `ObserverPingRetry` and `ObserverPingInterval` properties are set.

Category	Description
Datatype	Number
Valid values	>=10
Broker default	10 milliseconds

Category	Description
Imported?	No
Parameter class	Not applicable
Role	Primary
Standby type	Not applicable
Corresponds to	Not applicable
Scope	Broker configuration
Cloud Control name	Not applicable

ObserverReconnect

The `ObserverReconnect` configuration property specifies how often the observer establishes a new connection to the primary database.

When this property is set to the default value of 0, the observer creates and maintains a connection to the primary database, but it does not periodically create a new connection to the primary database. While this eliminates the processing overhead associated with periodically establishing a new observer connection to the primary database, it also prevents the observer from detecting that it is not possible to create new connections to the primary database. Oracle recommends that this property be set to a value that is small enough to allow timely detection of faults at the primary database, but large enough to limit the overhead associated with periodic observer connections to an acceptable level.

Category	Description
Datatype	Integer
Valid values	Integral number of seconds. Must be greater than, or equal to, 0.
Broker default	0
Imported?	No
Parameter class	Not applicable
Role	Primary and standby
Standby type	Not applicable
Corresponds to	Not applicable
Scope	Broker configuration. This property will be consumed by the observer after fast-start failover has been enabled.
Cloud Control name	Not applicable

OperationTimeout

The `OperationTimeout` configuration property specifies the maximum amount of time the broker should wait for health check, get monitorable property, and set property operations to complete.

Category	Description
Datatype	Integer
Valid values	>= 30 and <= 3600
Broker default	30 seconds
Imported?	No
Parameter class	Not applicable
Role	Primary and standby
Standby type	Not applicable
Corresponds to	Not applicable
Scope	Broker configuration. This property will be consumed by broker on all databases in the configuration.
Cloud Control name	Not applicable

PreferredObserverHosts

The `PreferredObserverHosts` configurable property allows you to list the hosts for the master observer when that database is in the primary role.

This property cannot be changed if Fast-Start failover is enabled and the master observer running.

If this property is set, then after a role change the broker takes the following steps:

1. Checks whether the current master observer is running on a host in the list specified by the `PreferredObserverHosts` property of the new primary. If yes, stop here; otherwise, go to step 2.
2. Scans the list from the beginning to look for a host where one observer is running, and connects to the primary and target standby databases. If found, go to step 3; otherwise the configuration will move to the unobserver state and there will be no master observer until an observer specified in this property becomes available.
3. Attempts to switch the master observer to the host found in step 2. If switching fails, no other hosts are tried.

Category	Description
Data type	String
Valid value	A character string that contains one or more host names, separated by comma: <code>host_name_1:priority [,host_name_n:priority]</code> Each host name can be any character string for a valid network name, even if there is no observer running on this host when it is set.
Broker default	Null
Imported?	No
Parameter class	Not applicable
Role	Primary, standby

Category	Description
Standby type	Physical standby and logical standby
Corresponds to	Not applicable
Scope	Database
Cloud Control name	Not applicable

PrimaryLostWriteAction

The `PrimaryLostWriteAction` configuration property determines what action is taken if a primary or standby database detects that a lost write has occurred at the primary database.

Note:

The `PrimaryLostWriteAction` configuration property is available only in Oracle Database 11g Release 2 (11.2.0.4) and in Oracle Database 12c Release 1 (12.1.0.2) and later.

The possible actions are as follows:

- `CONTINUE` - The primary database continues operating even if a primary or standby database detects that a lost write has occurred at the primary database. This is the default action.
- `SHUTDOWN` - If a primary or standby database detects that a lost write has occurred at the primary database, then fast-start failover is disabled and the primary database performs a shutdown abort. Automatic failover will not occur.
- `FAILOVER` - If fast-start failover is enabled in maximum performance mode, and the configuration is within the acceptable limit specified for the `FastStartFailoverLagLimit` property, then the observer initiates a failover.
- `FORCEFAILOVER` - If fast-start failover is enabled (in either maximum performance or maximum availability mode), then the observer initiates a failover. This option results in a data loss failover.

For both the `FAILOVER` and `FORCEFAILOVER` options, if fast-start failover is disabled then no failover occurs, but the primary is shut down.

If a primary or standby database detects that a lost write has occurred at the primary database, fast-start failover is disabled if any of the following conditions are true:

- `PrimaryLostWriteAction` is set to `CONTINUE`
- `PrimaryLostWriteAction` is set to `SHUTDOWN`
- `PrimaryLostWriteAction` is set to `FAILOVER` and protection mode is set to either maximum availability or maximum protection

Diagnostic information is written to the database alert and broker logs at the primary database and at the standby database where the lost write was detected.

If a lost write occurs at the primary database, then follow the guidelines in "Resolving ORA-752 or ORA-600 [3020] During Standby Recovery" in My Oracle Support Note 1265884.1 located at <http://support.oracle.com>.



Note:

The `DB_LOST_WRITE_PROTECT` database initialization parameter must be set to `TYPICAL` or `FULL` at the primary database and at each standby database in the configuration to ensure that lost primary writes can be detected by all standby databases in the configuration.

Category	Description
Datatype	String
Valid values	CONTINUE, SHUTDOWN, FAILOVER, or FORCEFAILOVER
Broker default	CONTINUE
Imported?	No
Parameter class	Not applicable
Role	Primary and standby
Standby type	Not applicable
Corresponds to	Not applicable
Scope	Broker configuration
Cloud Control name	Not applicable

TraceLevel

The `TraceLevel` configuration property is used to control the amount of tracing performed by the broker for every member in the configuration.

Setting the property to `USER` limits the tracing to completed operations and to any warning or error messages resulting from an operation or health check. Setting the property to `SUPPORT` increases the amount of tracing to include lower-level information needed by Oracle Support Services.

Category	Description
Datatype	String
Valid values	USER, SUPPORT
Broker default	USER
Imported?	No
Parameter class	Not applicable
Role	Primary and standby
Standby type	Not applicable
Corresponds to	Not applicable
Scope	Database

Category	Description
Cloud Control name	Not applicable

Monitorable (Read-Only) Properties

Monitorable properties allow you to view information related to the database, database instance, or far sync instance.

You cannot change the values of monitorable properties. You can view all of them using the DGMGRL `SHOW` commands.

Note:

Information for monitorable properties can be seen only when broker management of the entity is enabled. Cloud Control displays the information obtained from these properties on the Property page.

If the database is an Oracle RAC database, the output values of some properties may also show instance-specific information. For example if the primary database is an Oracle RAC database, `LogXptStatus` may show `Instance1` transmitting redo data to `Standby2` has an error and `Instance2` transmitting redo data to `Standby4` has an error.

The following sections describe the monitorable properties:

- [InconsistentLogXptProps \(Inconsistent Redo Transport Properties\)](#)
- [InconsistentProperties \(Inconsistent Properties\)](#)
- [LogXptStatus \(Redo Transport Status\)](#)
- [LsbyFailedTxnInfo \(Logical Standby Failed Transaction Information\)](#)
- [LsbyParameters \(Logical Standby Parameters\)](#)
- [RecvQEntries \(Receive Queue Entries\)](#)
- [SendQEntries \(Send Queue Entries\)](#)
- [TopWaitEvents](#)

InconsistentLogXptProps (Inconsistent Redo Transport Properties)

The `InconsistentLogXptProps` monitorable property returns a table that shows all properties related to redo transport services whose values are inconsistent between the broker configuration file and the runtime value.

Although the properties reported in this table are database-specific or far sync instance-specific properties, the inconsistency is reported on an instance-specific basis. A database or far sync instance-specific property only ensures that there is one value in the broker's configuration file for all instances of the database or far sync instance, but the runtime values can be different. This means that a database or far sync instance-specific property may be inconsistent only on some instances.

This property pertains to the primary database, a physical standby database that ships redo data, or a far sync instance. The table contains the following columns:

- `INSTANCE_NAME`
The value identifying the SID for the instance.
- `STANDBY_NAME`
The database unique name (`DB_UNIQUE_NAME`) of the standby database or far sync instance to which this redo transport services property pertains.
- `PROPERTY_NAME`
The name of the redo transport services property with an inconsistent value.
- `MEMORY_VALUE`
The runtime value being used in the database or far sync instance.
- `BROKER_VALUE`
The value of the redo transport services property saved in the broker configuration file.

InconsistentProperties (Inconsistent Properties)

The `InconsistentProperties` monitorable property shows all property values in the broker configuration file that are inconsistent with the values in the corresponding server parameter file or the runtime values.



Note:

The `InconsistentProperties` property is deprecated as of Oracle Database 19c and may be desupported in a future release. This property always has no value.

The information is shown in table format. Although the properties reported in this table are database-specific or far sync instance-specific properties, the inconsistency is reported on an instance-specific basis. A database or far sync instance-specific property only ensures that there is one value in the broker's configuration file for all instances of the database or far sync instance, but the runtime values can be different. This means that a database or far sync instance-specific property may be inconsistent only on some instances

Each individual configuration member has this property. The table contains the following columns:

- `INSTANCE_NAME`
The value identifying the SID for the instance.
- `PROPERTY_NAME`
The name of the property with the inconsistent value.
- `MEMORY_VALUE`
The corresponding runtime value being used.
- `SPFILE_VALUE`

The corresponding value saved in the server parameter file (SPFILE).

- `BROKER_VALUE`

The value of the property saved in the broker configuration file.

LogXptStatus (Redo Transport Status)

The `LogXptStatus` monitorable property returns a table that contains the error status of redo transport services for each of the enabled configuration members.

This property pertains to the primary database, a physical standby database that ships redo data, or a far sync instance.

The table contains the following columns:

- `PRIMARY_INSTANCE_NAME`

The value identifying the SID for the instance on the primary database.

- `STANDBY_DATABASE_NAME`

The database unique name (`DB_UNIQUE_NAME`) of the standby database or far sync instance.

- `ERROR`

The text of the redo transport error. If there is no error, the field is empty.

Each entry in the table indicates the status of redo transport services on one redo source to one redo destination.

The error status can be an empty string, which indicates there is no error.

In the following example, the `STATUS` from `South_Sales` is empty because there is no error for the `South_Sales` destination. The `South_Report` destination returned the `ORA-01034` message.

```
DGMGRL> SHOW DATABASE 'North_Sales' 'LogXptStatus' ;
LOG TRANSPORT STATUS
PRIMARY_INSTANCE_NAME STANDBY_DATABASE_NAME          STATUS
                    north_sales1          South_Sales
                    north_sales1          South_Report    ORA-01034: ORACLE not available
```

LsbyFailedTxnInfo (Logical Standby Failed Transaction Information)

The `LsbyFailedTxnInfo` monitorable property identifies a failed transaction that caused log apply services to stop.

This property contains a string with the following values from the `DBA_LOGSTDBY_EVENTS` view:

- `XIDUSN`: Transaction ID undo segment number
- `XIDSLT`: Transaction ID slot number
- `XIDSQN`: Transaction ID sequence number
- `STATUS_CODE`: Status (or Oracle error code) belonging to the `STATUS` message
- `STATUS`: Description of the current activity of the process or the reason why log apply services stopped

The transaction IDs and status information are separated by a string of number signs (###).
This property pertains to a logical standby database.

LsbyParameters (Logical Standby Parameters)

The `LsbyParameters` monitorable property contains a string that identifies the value of `MAX_SGA` and `MAX_SERVERS` specifically reserved for log apply services.

The `MAX_SGA` is the maximum system global area and `MAX_SERVERS` is the maximum number of parallel query servers. These values are separated by a string of number signs (###) in the `LsbyParameters` property.

This property pertains to a logical standby database.

Note:

As of Oracle Database Release 19c, the `LsbyMaxEventsRecorded` property is deprecated and may be desupported in a future release.

RecvQEntries (Receive Queue Entries)

The `RecvQEntries` monitorable property returns a table indicating all log files that were received by the standby database but have not yet been applied.

If no rows are returned, it implies all log files received have been applied. This property pertains to a standby database.

The table contains the following columns in the order shown:

- STATUS

The `STATUS` column is set to one of the following values for a log file on a logical standby database:

- `NOT_APPLIED`: No redo records in this log file have been applied.
- `PARTIALLY_APPLIED`: Some of the redo records in this log file have been applied while others have not.
- `COMMITTED_TRANSACTIONS_APPLIED`: This status value only applies to a logical standby database. All redo records belonging to the committed transactions have been applied. Redo records belonging to uncommitted transactions have not been read by LogMiner and may still be needed when the transactions are committed in the future. Therefore, it is not safe yet to discard this online redo log file.

- RESETLOGS_ID

Resetlogs identifier associated with the archived redo log file

- THREAD

The redo thread number

- LOG_SEQ

The online redo log file sequence number

- `TIME_GENERATED`
The first time when the online redo log file was written to the primary database
- `TIME_COMPLETED`
The next time when the log file was archived on the primary database (corresponds to the `NEXT_CHANGE#` column)
- `FIRST_CHANGE#`
First change number in the archived redo log file
- `NEXT_CHANGE#`
First change in the next log file
- `SIZE (KBs)`
The `SIZE` of the online redo log file in kilobytes

For example:

```
DGMGRL> SHOW DATABASE 'South_Sales' 'RecvQEntries' ;
          STATUS      RESETLOGS_ID      THREAD
LOG_SEQ   TIME_GENERATED    TIME_COMPLETED    FIRST_CHANGE#
NEXT_CHANGE#  SIZE (KBs)
          NOT_APPLIED      497198843          1              5
06/20/2003 14:55:38 06/20/2003 16:31:26      202138
210718      7364
          NOT_APPLIED      497198843          1              6
06/20/2003 16:31:26 06/20/2003 16:31:39      210718
210753      13
          NOT_APPLIED      497198843          1              7
06/20/2003 16:31:39 06/20/2003 16:31:54      210753
210758      1
          NOT_APPLIED      497198843          1              8
06/20/2003 16:31:54 06/20/2003 16:31:59      210758
210789      11
```



Note:

Cloud Control displays this information on the Log File Details page.

SendQEntries (Send Queue Entries)

The `SendQEntries` monitorable property returns a table that shows all log files on the primary database that were not successfully archived to one or more standby databases.

This property pertains to the primary database.

The table contains the following columns:

- `STANDBY_NAME`
The value can be empty or it can contain the database unique name (`DB_UNIQUE_NAME`) of a standby database. If empty, the `STATUS` column will contain a value of `CURRENT` or `NOT_ARCHIVED`.

- STATUS

The STATUS column is set to one of the following values:

- CURRENT: A log file to which online redo is currently being written.
- NOT_ARCHIVED: A completed online redo log file that has not been archived locally.
- ARCHIVED: A completed log file that has been archived locally but has not been transmitted to the standby database specified in the STANDBY_NAME column.

The table contains exactly one row with the value of STATUS=CURRENT. There can be multiple rows with the value STATUS=ARCHIVED or STATUS=NOT_ARCHIVED.

- RESETLOGS_ID

Resetlogs identifier associated with the archived redo log file

- THREAD

The redo thread number.

- LOG_SEQ

The log sequence number. Multiple rows may have the same LOG_SEQ value (for different STANDBY_NAME values).

- TIME_GENERATED

The first time when the online redo log file was written to the primary database.

- TIME_COMPLETED

The next time when the log file was archived on the primary database (corresponds to the NEXT_CHANGE# column).

- FIRST_CHANGE#

First change number in the archived redo log file.

- NEXT_CHANGE#

First change in the next log file.

- SIZE (KBs)

The SIZE of the online redo log file in kilobytes.

For example, the following shows output from a SHOW DATABASE command:

```
DGMGRL> SHOW DATABASE 'North_Sales' 'SendQEntries' ;
PRIMARY_SEND_QUEUE
      STANDBY_NAME      STATUS      RESETLOGS_ID
THREAD      LOG_SEQ      TIME_GENERATED      TIME_COMPLETED
FIRST_CHANGE#  NEXT_CHANGE#  SIZE (KBs)
      South_Sales  ARCHIVED      497198843
1
210789      9 06/20/2003 16:31:59 06/20/2003 16:39:57
211411      211411      186
      South_Sales  ARCHIVED      497198843
1
211411      10 06/20/2003 16:39:57 06/20/2003 16:40:01
211411      211415      1
      South_Sales  ARCHIVED      497198843
1
211415      11 06/20/2003 16:40:01 06/20/2003 16:40:07
211415      211418      1
      CURRENT      497198843
1
211418      12 06/20/2003 16:40:07
211418      1
```

**Note:**

Cloud Control displays this information on the Log File Details page.

TopWaitEvents

The `TopWaitEvents` monitorable property specifies the 5 events with the longest waiting time in the specified instance.

The events and their waiting time are retrieved from `V$SYSTEM_EVENT`. Each instance in the configuration has this property. This property is an instance-specific monitorable property. The table contains the following columns in the order shown:

- **Event**
The system wait event.
- **Wait Time**
The total amount of time waited for this event in hundredths of a second.

The following example shows output from a `SHOW INSTANCE` command:

```
DGMGRL> SHOW INSTANCE north_sales1 'TopWaitEvents';
TOP SYSTEM WAIT EVENTS
      Event                               Wait Time
-----
rdbms ipc message                        671350
SQL*Net message from client                62390
      pmon timer                           47897
Queue Monitor Wait                         43016
wakeup time manager                       38508
```

Configurable Properties

Configurable properties control the behavior of broker configuration members.

You can view and dynamically update the values of these properties using either DGMGRL or Cloud Control. However, some properties can only be updated through DGMGRL.

These properties control certain database initialization parameters and SQL statements that the broker uses to manage an Oracle Data Guard broker configuration. Therefore, you should not manually set those initialization parameters or issue those SQL statements on a broker configuration.

In most cases, the configurable property is said to apply to the entire member, meaning the value you set for the property applies to each instance of the member (that is, database or far sync instance). However, in a few cases, the configurable property is said to have instance-specific scope which means that for a multiple-instance database or far sync instance, it is possible that the values of some particular properties may differ between instances. The following table lists each configurable property and indicates whether the scope of the property is member-wide or instance-specific. If the `Scope` column contains:

- **Member**—The value of the property is database or far sync instance-wide, not instance or configuration specific.

- Instance—The value of the property is instance specific.
If there is an asterisk (*) present, it indicates that the property value can be set for all instances of an Oracle RAC database using the `EDIT INSTANCE * ON DATABASE` command.
- Configuration—The value of the property is configuration wide, not instance or member specific.

Table 11-1 Configurable Properties

Configurable Property Name	Scope	Pertains To
AlternateLocation	Member	Redo transport services
ArchiveLocation	Member	Redo transport services
ApplyInstances	Member	Redo Apply
ApplyInstanceTimeout	Member	Redo Apply and SQL Apply
ApplyLagThreshold	Member	Redo transport services
ApplyParallel	Member	Redo Apply
Binding	Member	Redo transport services
DelayMins	Member	Redo Apply and SQL Apply
DGConnectIdentifier	Member	Broker communication, Redo transport services
Encryption	Member	Redo transport services
FastStartFailoverTarget	Member	Fast-start failover
InstanceName	Instance	Instance identification
LogShipping	Member	Redo transport services
LogXptMode	Member	Redo transport services
MaxFailure	Member	Redo transport services
NetTimeout	Member	Redo transport services
ObserverConnectIdentifier	Member	Fast-start failover
PreferredApplyInstance	Member	Redo Apply and SQL Apply
PreferredObserverHosts	Member	Fast-start failover
RedoCompression	Member	Redo transport services
RedoRoutes	Member	Redo transport services
ReopenSecs	Member	Redo transport services
StandbyAlternateLocation	Member	Redo transport services
StandbyArchiveLocation	Member	Redo transport services
StaticConnectIdentifier	Instance	Instance Startup and Shutdown
TransportDisconnectedThreshold	Member	Redo transport services
TransportLagThreshold	Member	Redo transport services

 **Note:**

Starting with Oracle Database Release 19c, configurable properties that map directly to database initialization parameters and logical standby attributes are not maintained within the broker configuration file. See the "Deprecated Features" section in [Changes in Oracle Database Release 21c for Oracle Data Guard Broker](#) for a list of deprecated initialization parameters and logical standby attributes.

 **Note:**

When a broker configuration with its primary database is created and members are added to the configuration, the broker imports existing settings from the members to set many of the properties. If importing an existing setting fails, or if a property value is not imported, then the broker uses a broker default value. The default values and whether or not a property is imported is indicated within each property description.

 **See Also:**

[Managing the Members of a Broker Configuration](#) for more information about property management

Broker Controlled Database Initialization Parameters and SQL Statements

The following database initialization parameters are controlled by broker configurable properties. Therefore, you should not set these parameters manually:

- `LOG_ARCHIVE_DEST_n`
- `LOG_ARCHIVE_DEST_STATE_n`

The broker also uses configurable property settings to manage how apply is to be started. Therefore, the following SQL statements are managed automatically by the broker:

- `ALTER DATABASE RECOVER MANAGED STANDBY DATABASE`
- `ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE`

AlternateLocation

The `AlternateLocation` configurable property specifies an alternate online redo log archive location for primary, logical, and snapshot standby databases when the location specified by the `ArchiveLocation` configurable property fails.

If the `StandbyArchiveLocation` property is not empty, the `AlternateLocation` property specifies an alternate online redo log archive location. If the `StandbyArchiveLocation`

property is empty, the `AlternateLocation` property specifies an alternate online and standby redo log archive location.

This property has database-specific scope, and the location that it specifies is applicable to all instances in a database.

Category	Description
Datatype	String
Valid values	<ul style="list-style-type: none"> An empty string when no alternate location is desired. A directory specification that is accessible by the database. A valid <code>LOG_ARCHIVE_DEST_n</code> parameter string with the <code>LOCATION</code> attribute specified, but no <code>VALID_FOR</code>, <code>ALTERNATE</code>, or <code>SERVICE</code> attributes.
Broker default	Empty string
Imported?	No
Parameter class	Dynamic
Role	Primary and standby
Standby type	Physical, logical, or snapshot standby
Corresponds to	On the primary or standby database, the <code>LOCATION</code> attribute for the <code>LOG_ARCHIVE_DEST_n</code> initialization parameter that represents an alternate destination of the local destination that matches the configurable property <code>ArchiveLocation</code> .
Scope	Member
Cloud Control name	There is no Cloud Control name

ArchiveLocation

The `ArchiveLocation` configurable property specifies the online redo log archive location for primary, logical, and snapshot standby databases.

If the `StandbyArchiveLocation` property is not empty, the `ArchiveLocation` property specifies an online redo log archive location. If the `StandbyArchiveLocation` property is empty, the `ArchiveLocation` property specifies an online and standby redo log archive location.

Set a value for this property if you want Oracle to manage local archiving.

Category	Description
Datatype	String
Valid values	<ul style="list-style-type: none"> An empty string when you do not want broker to manage the online archive location. A nonempty file specification of the redo log archive location for the database. Specify <code>USE_DB_RECOVERY_FILE_DEST</code> if a database recovery area is desired. A valid <code>LOG_ARCHIVE_DEST_n</code> parameter string with the <code>LOCATION</code> attribute specified, but no <code>VALID_FOR</code>, <code>ALTERNATE</code>, or <code>SERVICE</code> attributes.
Broker default	Empty string

Category	Description
Imported?	No
Parameter class	Dynamic
Role	Primary and standby
Standby type	Physical, logical, or snapshot standby
Corresponds to	<ul style="list-style-type: none"> LOCATION attribute of the LOG_ARCHIVE_DEST_n initialization parameter of the primary or standby database with VALID_FOR compatible with (ONLINE_LOGFILE, ALL_ROLES) DESTINATION column of the V\$ARCHIVE_DEST view of the primary or standby database
Scope	Member
Cloud Control name	There is no Cloud Control name

ApplyInstances

The `ApplyInstances` property lets you specify how many physical standby instances run Redo Apply.

As of Oracle Database 12c Release 2 (12.2.0.1), a new `INSTANCES` keyword is available on the `SQL ALTER RECOVER MANAGED STANDBY DATABASE` statement (only for Oracle Real Application Clusters (Oracle RAC) or Oracle RAC One Node databases). When used, it causes Redo Apply to run on each active running physical standby instance that is running in the same mode as the instance on which Redo Apply was started. You can specify `ALL` or you can specify a specific number to restrict the number of instances that Redo Apply uses. If a database has already been set up to run Redo Apply on multiple instances, then you can use the Data Guard broker property `ApplyInstances` to restrict the number of instances that are involved in Redo Apply on an Oracle RAC physical standby database.



See Also:

- *Oracle Data Guard Concepts and Administration* for more information about setting up multi-instance Redo Apply.

Category	Description
Datatype	Integer
Valid values	0 to 16, ALL <ul style="list-style-type: none"> • ALL: indicates that redo apply will be started on all currently active instances • 0 and 1: recovery will be started in single instance mode. • >1: indicates that redo apply should not run on any more than the number of instances specified.

Category	Description
Broker default	The default is 0.
Imported?	Yes
Parameter class	Not applicable
Role	Standby
Standby type	Physical standby
Corresponds to	INSTANCES keyword of ALTER DATABASE RECOVER MANAGED STANDBY DATABASE command.
Scope	Database
Cloud Control name	Not applicable

ApplyInstanceTimeout

The `ApplyInstanceTimeout` configurable property specifies the number of seconds the broker waits after detecting the current apply instance failed before initiating the apply instance failover.

Category	Description
Datatype	Integer
Valid values	≥ 0 (seconds)
Broker default	0 (results in immediate apply instance failover)
Imported?	No
Parameter class	Not applicable
Role	Standby, Recovery Appliance
Standby type	Physical, logical, Recovery Appliance
Corresponds to	Not applicable
Scope	Database
Cloud Control name	Not applicable

ApplyLagThreshold

The `ApplyLagThreshold` configurable property generates a warning status for a logical or physical standby when the member's apply lag exceeds the value specified by the property.

The property value is expressed in seconds. A value of 0 seconds results in no warnings being generated when an apply lag exists.

Category	Description
Datatype	Number
Valid values	≥ 0
Broker default	30 seconds
Imported?	No
Parameter class	Not applicable

Category	Description
Role	Standby, Recovery Appliance
Standby type	Physical, logical
Corresponds to	Not applicable
Scope	Database
Cloud Control name	Not applicable

ApplyParallel

The `ApplyParallel` configurable property specifies whether Redo Apply should use multiple processes to apply redo data to the physical standby database.

If Redo Apply is shut off, then setting the property has no immediate effect. However, when Redo Apply is running again, the value of the property is used to determine the parallel apply behavior of Redo Apply.

Category	Description
Datatype	String
Valid values	<ul style="list-style-type: none"> <code>AUTO</code>—the number of parallel processes used for Redo Apply is automatically determined by Oracle based on the number of CPUs that the system has. <code>NO</code>—no parallel apply 2, 3, and so on—manually specify the number of parallel processes used for Redo Apply. (Specifying 0 is the same as specifying <code>NO</code>; specifying 1 is the same as specifying <code>AUTO</code>.)
Broker default	<code>AUTO</code>
Imported?	No
Parameter class	Not applicable
Role	Standby
Standby type	Physical
Corresponds to	<ul style="list-style-type: none"> <code>AUTO</code> corresponds to the <code>PARALLEL</code> clause of the <code>ALTER DATABASE RECOVER MANAGED STANDBY DATABASE</code> statement <code>NO</code> corresponds to the <code>NOPARALLEL</code> clause of the <code>ALTER DATABASE RECOVER MANAGED STANDBY DATABASE</code> statement 2, 3, and so on corresponds to the <code>PARALLEL n</code> clause of the <code>ALTER DATABASE RECOVER MANAGED STANDBY DATABASE</code> statement
Scope	Database
Cloud Control name	Not applicable

Binding

The `Binding` configurable property specifies whether the destination is `MANDATORY` or `OPTIONAL`.

Category	Description
Datatype	String
Valid values	<ul style="list-style-type: none"> MANDATORY You can specify a policy for reuse of online redo log files using the MANDATORY value. If the archiving operation of a mandatory destination fails, online redo log files cannot be overwritten. OPTIONAL You can specify a policy for reuse of online redo log files using the OPTIONAL value. If the archiving operation of an optional destination fails, the online redo log files are overwritten.
Broker default	OPTIONAL
Imported?	No
Parameter class	Dynamic
Role	Standby database, Recovery Appliance, far sync instance ¹
Standby type	Physical, logical, or snapshot standby, far sync instance, Recovery Appliance
Corresponds to	<ul style="list-style-type: none"> MANDATORY and OPTIONAL attributes for the LOG_ARCHIVE_DEST_n initialization parameter of the database or far sync instance that is sending redo data BINDING column of the V\$ARCHIVE_DEST view of the database or far sync instance that is sending redo data
Scope	Member
Cloud Control name	Not applicable

¹ Although this property is set for the redo destination, it is indirectly related to the redo transport services for the database or far sync instance that is sending redo data. The broker propagates the setting you specify to the corresponding attributes of the LOG_ARCHIVE_DEST_n value of the database or far sync instance that is sending redo data.

DelayMins

The `DelayMins` configurable property specifies the number of minutes log apply services will delay applying the archived redo log data on the standby database.

When the `DelayMins` property is set to the default value of 0 minutes, log apply services apply redo data as soon as possible.

If the `DelayMins` property is set to 0, start log apply services as follows:

- Start Redo Apply on physical standby databases using the following SQL statement:

```
ALTER DATABASE RECOVER MANAGED STANDBY DATABASE;
```

- Start SQL Apply on logical standby databases using the following SQL statement:

```
ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE;
```

Category	Description
Datatype	Integer
Valid values	≥ 0 (minutes) ¹
Broker default	0

Category	Description
Imported?	No
Parameter class	Dynamic
Role	Standby ² , Recovery Appliance
Standby type	Physical, logical, Recovery Appliance
Corresponds to	<ul style="list-style-type: none"> • DELAY attribute for the LOG_ARCHIVE_DEST_n initialization parameter of the database or far sync instance that is sending redo data • DELAY_MINS column of the V\$ARCHIVE_DEST view of the database or far sync instance that is sending redo data • Options used to start Redo Apply and SQL Apply
Scope	Database
Cloud Control name	Apply Delay (mins)

- ¹ The DelayMins property must be reset to 0 before attempting a switchover to a standby with DelayMins greater than 0.
- ² Although this property is set for the standby database, it is indirectly related to the redo transport services for the database or far sync instance that is sending redo data. The broker propagates the setting you specify on the standby database to the corresponding attributes of the LOG_ARCHIVE_DEST_n value of the database or far sync instance that is sending redo data.

DGConnectIdentifier

The `DGConnectIdentifier` configurable property specifies the connection identifier the broker uses when making connections to a configuration member.

If you are using DGMGRL, then you supply the value when you enter the `CREATE CONFIGURATION`, `ADD DATABASE`, or `ADD FAR_SYNC` command. If you are using Cloud Control, the value is supplied automatically. The connect identifier for a configuration member must:

- Allow all other members in the configuration to reach it.
- Allow the member to reach itself.
- Allow all instances of an Oracle RAC database to be reached.
- Specify a service that all instances dynamically register with the listeners so that connect-time failover on an Oracle RAC database is possible.

Caution:

The service should NOT be one that is defined and managed by Oracle Clusterware.

- Have failover attributes set to allow the primary database's Redo Transport Services to continue shipping redo data to an Oracle RAC standby database, even if the receiving instance of that standby database has failed.

The value of this property is specified in the `SERVICE` attribute of the `LOG_ARCHIVE_DEST_n` parameter when the broker configures redo transport services on the primary database.

Category	Description
Datatype	String
Valid values	A connect identifier that can be used to connect to this database
Broker default	Not applicable
Imported?	No
Parameter class	Not applicable
Role	Primary, standby, far sync instance, Recovery Appliance
Standby type	Physical, logical, or snapshot standby, far sync instance, Recovery Appliance
Corresponds to	<code>SERVICE_NAME</code> attribute of the <code>LOG_ARCHIVE_DEST_n</code> initialization parameter of the configuration member that is sending redo data
Scope	Database, far sync instance, Recovery Appliance
Cloud Control name	Not applicable

 **See Also:**

Oracle Database Net Services Administrator's Guide

Encryption

The `Encryption` configurable property is used to specify whether redo data is encrypted before transmitting it to a Recovery Appliance.

 **Note:**

Redo transport encryption can only be used with a Recovery Appliance.

Category	Description
Datatype	String
Valid values	<code>DISABLE</code> , <code>ENABLE</code> <ul style="list-style-type: none"> When <code>ENABLE</code> is specified, the redo data is encrypted before it is sent to a Recovery Appliance. When <code>DISABLE</code> is specified, the redo data is not encrypted before it is sent to a Recovery Appliance.
Broker Default	<code>DISABLE</code>
Imported?	No
Parameter class	Dynamic
Role	Recovery Appliance
Standby type	Recovery Appliance
Corresponds to	<ul style="list-style-type: none"> <code>ENCRYPTION</code> attribute for the <code>LOG_ARCHIVE_DEST_n</code> initialization parameter of the database or far sync instance that is sending redo data <code>ENCRYPTION</code> column of the <code>V\$ARCHIVE_DEST</code> view of the database or far sync instance that is sending redo data to the Recovery Appliance

Category	Description
Scope	Recovery Appliance
Cloud Control name	Not Applicable

FastStartFailoverTarget

The `FastStartFailoverTarget` configuration property specifies the `DB_UNIQUE_NAME` of one or more standby databases that can act as target databases in a fast-start failover situation when the database on which the property is set is the primary database.

These possible target databases are referred to as candidate fast-start failover targets. See Task 2 in [Enabling Fast-Start Failover](#) for more information about setting this property.

The `FastStartFailoverTarget` configuration property can only be set to the name of physical or logical standbys. It cannot be set to the name of a snapshot standby database, far sync instance, or Zero Data Loss Recovery Appliance.

Category	Description
Datatype	String
Valid value	<code>DB_UNIQUE_NAME</code> of the database that is the target of the fast-start failover. You can specify names of multiple standbys or the keyword <code>ANY</code> . If multiple names are specified, then the broker selects them in the order in which they are listed. If the property is set to <code>ANY</code> , then the broker can select any one of all viable standbys of the primary as the current FSFO target.
Broker default	If only one physical or logical standby database exists, then the broker selects that as the default value for this property on the primary database when fast-start failover is enabled. If more than one physical or logical standby database exists, then the broker selects one based on the order in which they are specified in the property definition. Targets are verified when fast-start failover is enabled. For the target standby database, the broker automatically selects the current primary database as the value for this property when fast-start failover is enabled.
Imported?	No
Parameter class	Not applicable
Role	Primary or standby
Standby type	Physical or logical
Corresponds to	Not applicable
Scope	Database
Cloud Control name	Cloud Control displays the value for the current primary database on the Oracle Data Guard Overview page, along with whether or not fast-start failover has been enabled.

InstanceName

The `InstanceName` configurable property corresponds to the `INSTANCE_NAME` initialization parameter.

The property can only be updated when broker management of the database is disabled. You should only update the value when the `INSTANCE_NAME` initialization parameter is changed. In such a case, you must disable broker management of the database, update the `InstanceName` property to match the new `INSTANCE_NAME` value, and then reenable broker management of the database.

Note:

If no existing instance in the broker configuration, for the database whose name matches this instance's `db_unique_name`, has an `InstanceName` property with a value that matches the `INSTANCE_NAME` initialization parameter for this instance, then the broker creates a new instance for this database that has this instance's `db_unique_name`.

Category	Description
Datatype	String
Valid values	The value of the <code>INSTANCE_NAME</code> initialization parameter for this instance. On certain platforms, the value may be case-sensitive.
Broker default	Not applicable
Imported?	Yes
Parameter class	Not applicable
Role	Primary and standby
Standby type	Physical, logical, or snapshot
Corresponds to	<code>INSTANCE_NAME</code> initialization parameter
Scope	Instance
Cloud Control name	Not applicable

LogShipping

The broker uses the value of the `LogShipping` property when the primary database is in the `TRANSPORT-ON` state or when the physical standby or far sync instance forwards redo data to another member.

The other member can be a physical, logical, or snapshot standby, or a far sync instance.

- If the primary database is in the `TRANSPORT-ON` state and the value of the `LogShipping` property is `ON`, then redo transport services are enabled to send redo data to the particular configuration member. If the `LogShipping` property is `OFF`, then redo transport services are disabled to that member.
- If a configuration member that forwards redo data has its `LogShipping` property set to `ON` and the member to which it sends redo data also has its `LogShipping` property set to `ON`,

then redo transport services are enabled from the member sending redo data to the member that is to receive redo data.

If a member that forwards redo data has its `LogShipping` property set to `ON` but the member to which it sends redo data has its `LogShipping` property set to `OFF`, then redo transport services are disabled from the member sending redo data to the member that is to receive redo data.

Category	Description
Datatype	String
Valid values	ON or OFF
Broker default	ON
Imported?	No
Parameter class	Dynamic
Role	Standby database, Recovery Appliance, far sync instance ¹
Standby type	Physical, logical, or snapshot standby, far sync instance, Recovery Appliance
Corresponds to	ENABLE and DEFER values for the <code>LOG_ARCHIVE_DEST_STATE_n</code> initialization parameter of the database or far sync instance that is sending redo data
Scope	Database, Recovery Appliance
Cloud Control name	Log Shipping

¹ Although this property is set for a standby database or far sync instance, it is indirectly related to the redo transport services for the database or far sync instance that is sending redo data. The broker propagates the setting you specify on the standby database to the corresponding attributes of the `LOG_ARCHIVE_DEST_n` value of the database or far sync instance that is sending redo data.

LogXptMode

The `LogXptMode` configurable property enables you to set the redo transport service.

You set the redo transport services on each configuration member to one of the following modes:

- `SYNC`
Configures redo transport services for this configuration member using the `SYNC` and `AFFIRM` attributes of the `LOG_ARCHIVE_DEST_n` initialization parameter. Standby redo log files are required. This mode is required for the maximum protection or maximum availability data protection modes. This redo transport service enables the highest grade of data protection to the primary database, but also incurs the highest performance impact.
- `ASync`
Configures redo transport services for this configuration member using the `ASync` and `NOAFFIRM` attributes of the `LOG_ARCHIVE_DEST_n` initialization parameter. Standby redo log files are required. This mode enables a moderate grade of data protection to the primary database, and incurs a lower performance impact than `SYNC`.
- `FASTSync`

Configures redo transport services for this configuration member using the `SYNC` and `NOAFFIRM` attributes of the `LOG_ARCHIVE_DEST_n` initialization parameter. This mode is only available in maximum availability protection mode. Because `FASTSYNC` transport mode uses the `NOAFFIRM` attribute of the `LOG_ARCHIVE_DEST_n` parameter, data loss is possible.

Category	Description
Datatype	String
Valid values	<code>SYNC</code> , <code>ASYNC</code> , or <code>FASTSYNC</code>
Broker default	<code>ASYNC</code>
Imported?	No
Parameter class	Dynamic
Role	Standby database, Recovery Appliance, far sync instance ¹
Standby type	Physical, logical, or snapshot standby, far sync instance, Recovery Appliance
Corresponds to	<ul style="list-style-type: none"> <code>SYNC</code>, <code>ASYNC</code>, <code>AFFIRM</code>, and <code>NOAFFIRM</code> attributes for the <code>LOG_ARCHIVE_DEST_n</code> initialization parameter of the database or far sync instance that is sending redo data <code>ARCHIVER</code>, <code>TRANSMIT_MODE</code>, and <code>AFFIRM</code> columns of <code>V\$ARCHIVE_DEST</code> view of the database or far sync instance that is sending redo data
Scope	Database, far sync instance, Recovery Appliance
Cloud Control name	Redo Transport Service

- ¹ Although this property is set for the standby database or far sync instance, it is indirectly related to the redo transport services for the database or far sync instance that is sending redo. The broker propagates the setting you specify on the standby database or far sync instance to the corresponding attributes of the `LOG_ARCHIVE_DEST_n` value of the database or far sync instance that is sending redo data. Note that if a database receives redo from a database or far sync instance where the `RedoRoutes` property has been configured with a redo transport mode, then the mode specified by that `RedoRoutes` property value overrides the value of the `LogXptMode` property.



See Also:

[Managing the Members of a Broker Configuration](#) for more information about setting data protection modes for redo transport services

MaxFailure

The `MaxFailure` configurable property specifies the maximum number of contiguous archiving failures before the redo transport services stop trying to transport archived redo log files to the standby database.

A value of zero indicates that an unlimited number of failures are allowed.

Category	Description
Datatype	Integer
Valid values	≥ 0

Category	Description
Broker default	If the standby database is part of a group as specified with the broker <code>RedoRoutes</code> property, then the default value is 1. Otherwise, the default value is 0.
Imported?	No
Parameter class	Dynamic
Role	Standby ¹ , Recovery Appliance
Standby type	Physical, logical, snapshot, Recovery Appliance
Corresponds to	<ul style="list-style-type: none"> <code>MAX_FAILURE</code> attribute for the <code>LOG_ARCHIVE_DEST_n</code> initialization parameter of the database or far sync instance that is sending redo data <code>MAX_FAILURE</code> column of the <code>V\$ARCHIVE_DEST</code> view of the database or far sync instance that is sending redo data
Scope	Database, Recovery Appliance
Cloud Control name	Not applicable

¹ Although this property is set for a standby database or far sync instance, it is indirectly related to the redo transport services for the database or far sync instance that is sending redo data. The broker propagates the setting you specify on the standby database or far sync instance to the corresponding attributes of the `LOG_ARCHIVE_DEST_n` value of the database or far sync instance that is sending redo data.

NetTimeout

The `NetTimeout` configurable property specifies the number of seconds the LGWR waits for Oracle Net Services to respond to a LGWR request.

It is used to bypass the long connection timeout in TCP.

Category	Description
Datatype	Integer
Valid values	0 to 1200
Broker default	30
Imported?	No
Parameter class	Dynamic
Role	Primary, standby, far sync instance, Recovery Appliance
Standby type	Physical, logical, or snapshot standby, far sync instance, Recovery Appliance
Corresponds to	<ul style="list-style-type: none"> <code>NET_TIMEOUT</code> attribute of the <code>LOG_ARCHIVE_DEST_n</code> initialization parameter of the database or far sync instance that is sending redo data <code>NET_TIMEOUT</code> column of <code>V\$ARCHIVE_DEST</code> view of the database or far sync instance that is sending redo data
Scope	Database, far sync instance, Recovery Appliance
Cloud Control name	Not applicable

ObserverConnectIdentifier

The `ObserverConnectIdentifier` configurable property specifies a connect identifier that can be used by the observer to connect to this database.

The connect identifier can pertain only to the primary database, or to the target standby database when fast-start failover is enabled.

Category	Description
Datatype	String
Valid Values	A connect identifier that observer can use to connect to this database
Broker Default	Empty String ¹
Imported?	No
Parameter Class	Not applicable
Role	Primary and Standby
Standby Type	Physical or logical
Corresponds to	Not applicable
Scope	Database
Cloud Control name	<code>Observer Connect Identifier</code>

¹ When this is Empty String (not set by the user), the connect identifier specified by this database's `DGConnectIdentifier` property will be used by the observer.

PreferredApplyInstance

The `PreferredApplyInstance` configurable property indicates that a particular instance is the preferred choice for serving log apply services.

It is only used when the database is a standby Oracle RAC database. The value could be an empty string (default) which means the broker chooses the apply instance.

Category	Description
Datatype	String
Valid Values	The instance name (SID) or empty string. Note that on certain platforms, SIDs may be case-sensitive.
Broker Default	Empty string
Imported?	No
Parameter Class	Not applicable
Role	Standby, Recovery Appliance
Standby Type	Physical, logical, Recovery Appliance
Corresponds to	Not applicable
Scope	Database
Cloud Control name	<code>Apply Instance</code>

 **See Also:**

- *Oracle Database Administrator's Guide* for more information about SIDs
- [Apply Services in an Oracle RAC Database Environment](#) for information about apply services in an Oracle RAC environment

PreferredObserverHosts

The `PreferredObserverHosts` configurable property allows you to list the hosts for the master observer when that database is in the primary role.

This property cannot be changed if Fast-Start failover is enabled and the master observer running.

If this property is set, then after a role change the broker takes the following steps:

1. Checks whether the current master observer is running on a host in the list specified by the `PreferredObserverHosts` property of the new primary. If yes, stop here; otherwise, go to step 2.
2. Scans the list from the beginning to look for a host where one observer is running, and connects to the primary and target standby databases. If found, go to step 3; otherwise the configuration will move to the unobserver state and there will be no master observer until an observer specified in this property becomes available.
3. Attempts to switch the master observer to the host found in step 2. If switching fails, no other hosts are tried.

Category	Description
Data type	String
Valid value	<p>A character string that contains one or more host names, separated by comma:</p> <pre>host_name_1:priority [,host_name_n:priority]</pre> <p>Each host name can be any character string for a valid network name, even if there is no observer running on this host when it is set.</p>
Broker default	Null
Imported?	No
Parameter class	Not applicable
Role	Primary, standby
Standby type	Physical standby and logical standby
Corresponds to	Not applicable
Scope	Database
Cloud Control name	Not applicable

RedoCompression

The `RedoCompression` configurable property is used to specify whether redo data is transmitted to a standby database or far sync instance in compressed or uncompressed form.



Note:

Redo transport compression is a feature of the Oracle Advanced Compression option. You must purchase a license for this option before using the redo transport compression feature.

Category	Description
Datatype	String
Valid value	DISABLE, ENABLE, ZLIB, or LZO <ul style="list-style-type: none"> When ENABLE is specified, the default compression algorithm is ZLIB. LZO is not supported when <code>LogXptMode</code> is set to SYNC or FASTSYNC.
Broker Default	DISABLE
Imported?	No
Parameter class	Dynamic
Role	Standby database, far sync instance ¹ , Recovery Appliance
Standby type	Physical, logical, or snapshot standby, far sync instance, Recovery Appliance
Corresponds to	<ul style="list-style-type: none"> COMPRESSION attribute for the <code>LOG_ARCHIVE_DEST_n</code> initialization parameter of the database or far sync instance that is sending redo data COMPRESSION column of the <code>V\$ARCHIVE_DEST</code> view of the database or far sync instance that is sending redo data
Scope	Database, far sync instance, Recovery Appliance
Cloud Control name	Not applicable

¹ Although this property is set for a standby database or far sync instance, it is indirectly related to the redo transport services for the database or far sync instance that is sending redo data. The broker propagates the setting you specify on the standby database or far sync instance to the corresponding attributes of the `LOG_ARCHIVE_DEST_n` value of the database or far sync instance that is sending redo data.

RedoRoutes

The `RedoRoutes` property lets you override the default behavior by which a primary database sends the redo that it generates to every other redo transport destination in the configuration.

You can use the `RedoRoutes` property to create a more complex redo transport topology, such as one in which a physical standby database or a far sync instance forwards redo received from the primary database to one or more destinations, or one in which the redo transport mode used for a given destination is dependent on which database is in the primary role.

The `RedoRoutes` property is set to a character string that contains one or more redo routing rules. Each rule contains one or more redo sources and one or more redo destinations. A redo routing rule becomes active when one of the redo sources in the rule is in the primary

role. This results in redo from the primary database being sent to every redo destination in that rule.

Category	Description
Datatype	String
Valid values	A character string that contains one or more redo routing rules, each contained within a pair of parentheses: <code>(redo_routing_rule_1) [(redo_routing_rule_n)]</code> See " Redo Routing Rules " for more information about redo routing rules.
Broker default	Null
Imported?	No
Parameter class	Not applicable
Role	Primary, standby, far sync instance
Standby type	Physical standby and far sync instance
Corresponds to	<code>LOG_ARCHIVE_DEST_n</code>
Scope	Database, far sync instance
Cloud Control Name	RedoRoutes

Redo Routing Rules

The `RedoRoutes` property is set to a character string that contains one or more redo routing rules.

Each rule is contained within a set of parentheses, as follows:

```
(redo_routing_rule_1) [(redo_routing_rule_n)]
```

A redo routing rule contains a redo source field and a redo destination field separated by a colon:

```
(redo source : redo destination)
```

The redo source field must contain the keyword `LOCAL` or `ANY`, or a comma-separated list of `DB_UNIQUE_NAME` values, as follows:

```
{LOCAL | ANY | db_unique_name_1 [, db_unique_name_n]}
```

- The `LOCAL` keyword is an alias for the local database name. This keyword cannot be used at a far sync instance.
- The `ANY` keyword is an alias for any database in the configuration.
- A database cannot be specified as a redo source in more than one redo routing rule defined at a given database, either explicitly or implicitly through use of the `LOCAL` keyword.

The redo destination field must contain the keyword `ALL` or a comma-separated list of redo destination groups, each of which consists of destination databases with optional priority attributes and optional redo transport mode attributes:

```
{ALL [xpt_mode] | redo_dest_group_1 [, redo_dest_group_n]}
```

- The `ALL` keyword is an alias for all possible destinations in the configuration.

- The `redo_dest_group_n` is:

```
{ db_unique_name_1 [xpt_mode] | ( db_unique_name_1 [xpt_mode]
[PRIORITY=n] [,db_unique_name_n [xpt_mode] [PRIORITY=n]] ) }
```

The optional `xpt_mode` specifies the redo transport mode to be used to send redo to the associated destination. It can have one of three values: `ASync`, `Sync`, or `FASTSync`. If the redo transport attribute is not specified, then the transport mode used will be the one specified by the `LogXptMode` property for the redo destination.

The optional `[PRIORITY=n]` can have $n = 1 \sim 8$. The default value for far sync members is 1 and the default value for non-far sync members is 8. It is important to understand how different group and priority settings affect redo transport under various conditions. The following examples describe some sample situations.

Example 1: Different Priorities Within a Group

Assume there are three members in a broker configuration:

```
PRI (primary database)
SB1 (standby database)
FS1 (far sync instance)
```

The primary database `PRI` generates the redo log for use by the standby database `SB1`. Because there is a far sync instance, `FS1`, there are two possible redo delivery paths to the standby database:

```
(path 1) PRI → FS1 → SB1
(path 2) PRI → SB1
```

Assume (path 1) is more desirable than (path 2). This can be expressed with the `RedoRoutes` property as follows:

```
PRI —RedoRoutes = (local : ( FS1 PRIORITY=1, SB1 PRIORITY=2 ) )
FS1 —RedoRoutes = ( PRI : SB1 )
```

As specified in the `PRI RedoRoutes` property, the primary (`PRI`) has two destinations : one for `FS1` with `PRIORITY=1` and one for `SB1` with `PRIORITY=2`. Smaller priority numbers mean higher priority, so primary `PRI` tries to ship the redo log to `FS1` first.

If `FS1` is available, then the primary ships to `FS1` which has `PRIORITY=1`. If `FS1` is unavailable, then the primary ships to `SB1` which has `PRIORITY=2`. When `FS1` becomes active again, the primary will ship to it again because priority 1 is higher than priority 2.

Example 2: The Same Priorities Within a Group

Suppose you add one more far sync instance, `FS2`, to the configuration set up in Example 1, and then update the `RedoRoutes` property as follows:

```
PRI — RedoRoutes = (local : ( FS1 PRIORITY=1, FS2 PRIORITY=1 ) )
FS1 — RedoRoutes = ( PRI : SB1 )
FS2 — RedoRoutes = ( PRI : SB1 )
```

The primary `PRI` now has two destinations, `FS1` and `FS2`, with the same priority. The primary must choose either `FS1` or `FS2`. Assume the primary chooses `FS1`.

If FS1 is available, then the primary ships to FS1. If FS1 is unavailable, then the primary ships to FS2. Even after FS1 becomes active again, the primary continues shipping to FS2 because FS1 and FS2 have the same priority. If FS2 fails, then the primary ships to FS1.

Example 3: Multiple Groups

Assume that you add one more standby, SB2, to the configuration, and then update the RedoRoutes property as follows so that the primary has two destination groups:

```
PRI — RedoRoutes = (local : ( FS1 PRIORITY=1, SB1 PRIORITY=2 ), ( FS2
PRIORITY=1, SB2 PRIORITY=2 ) )
FS1 — RedoRoutes = ( PRI : SB1 )
FS2 — RedoRoutes = ( PRI : SB2 )
```

The general rule is that there is one active redo path for each group. (See Example 4 for a scenario in which there are multiple active redo paths.) The primary establishes one redo delivery path for the first group (FS1 PRIORITY=1, SB1 PRIORITY=2), and establishes another redo delivery path for the second group (FS2 PRIORITY=1, SB2 PRIORITY=2).

- If both FS1 and FS2 are available, then the primary ships to FS1 and FS2.
- If FS1 is unavailable and FS2 is available, then the primary ships to SB1 and FS2.
- If FS1 is available and FS2 is unavailable, then the primary ships to FS1 and SB2.
- If both FS1 and FS2 are unavailable, then the primary ships to SB1 and SB2.

Example 4: The PRIORITY Attribute is Set to 8

Setting PRIORITY=8 has special meaning. If a primary ships redo to a destination with PRIORITY=8, then it must ship to every PRIORITY=8 destination. Suppose you update the RedoRoutes property as follows so that the primary has one group containing three destinations:

```
PRI — RedoRoutes = (local : ( FS1 PRIORITY=1, SB1 PRIORITY=8, SB2
PRIORITY=8 ) )
FS1 — RedoRoutes = ( PRI : SB1, SB2 )
```

These settings result in the following behavior:

- If FS1 is available, then the primary ships to FS1.
- If FS1 is unavailable, then the primary ships to both SB1 and SB2 because they both have PRIORITY=8
- If FS1 becomes active again, then the primary ships to FS1.

Usage Notes for Advanced Redo Transport Settings

The following usage notes apply regarding advanced redo transport settings:

- The RedoRoutes property has a default value of NULL, which is treated as (LOCAL : ALL) at a primary database.
- A redo routing rule is active if its redo source field specifies the current primary database. If a rule is active, primary database redo is sent by the database at which the rule is defined to each destination specified in the redo destination field of that rule.

- The `ASYNC` redo transport attribute must be explicitly specified for a cascaded destination to enable real-time cascading to that destination.
- The `RedoRoutes` property cannot be configured such that when a physical standby database is converted to a snapshot standby, the snapshot standby would send redo data to another member.
- The `RedoRoutes` property can be set for a logical standby database only if the redo source field is set to `LOCAL`.

ReopenSecs

The `ReopenSecs` configurable property specifies the minimum number of seconds before the archiver process (`ARCn`, foreground, or log writer process) should try again to access a previously failed destination.

Category	Description
Datatype	Integer
Valid values	>=0 seconds
Broker default	300
Imported?	No
Parameter class	Dynamic
Role	Standby database, far sync instance ¹ , Recovery Appliance
Standby type	Physical, logical, snapshot standby, far sync instance, Recovery Appliance
Corresponds to	<ul style="list-style-type: none"> • <code>REOPEN</code> attribute for the <code>LOG_ARCHIVE_DEST_n</code> initialization parameter of the database or far sync instance that is sending redo data • <code>REOPEN_SECS</code> column of the <code>V\$ARCHIVE_DEST</code> view of the database or far sync instance that is sending redo data
Scope	Database, far sync instance, Recovery Appliance
Cloud Control name	Not applicable

¹ Although this property is set for a standby database or far sync instance, it is indirectly related to the redo transport services for the database or far sync instance that is sending redo data. The broker propagates the setting you specify on the standby database or far sync instance to the corresponding attributes of the `LOG_ARCHIVE_DEST_n` value of the database or far sync instance that is sending redo data.

StandbyAlternateLocation

The `StandbyAlternateLocation` configurable property specifies an alternate standby redo log archive location to use when the location specified by the `StandbyArchiveLocation` configurable property fails.

The property has database-specific scope, and the location that it specifies is applicable to all instances in a database.

Category	Description
Datatype	String

Category	Description
Valid values	<ul style="list-style-type: none"> An empty string, when no alternate location is desired. A directory specification that is accessible by the instance. A valid <code>LOG_ARCHIVE_DEST_n</code> parameter string with the <code>LOCATION</code> attribute specified, but no <code>VALID_FOR</code>, <code>ALTERNATE</code>, or <code>SERVICE</code> attributes.
Broker default	Empty string
Imported?	No
Parameter class	Dynamic
Role	Primary, standby, and far sync instance
Standby type	Physical, logical, or snapshot standby, or a far sync instance
Corresponds to	On the standby database or the far sync instance, the <code>LOCATION</code> attribute for the <code>LOG_ARCHIVE_DEST_n</code> initialization parameter that represents an alternate destination of the local destination that matches the configurable property <code>StandbyArchiveLocation</code> .
Scope	Member
Cloud Control name	Alternate Standby Location



Note:

On a logical standby database, Oracle recommends the `LOCATION` attribute of the `LOG_ARCHIVE_DEST_n` initialization parameter for the local destination be different from the value of `StandbyAlternateLocation` configurable property.

StandbyArchiveLocation

The `StandbyArchiveLocation` configurable property specifies the standby redo log archive location. Oracle recommends that you always explicitly set the value.

Category	Description
Datatype	String
Valid values	<ul style="list-style-type: none"> An empty string when you do not want broker to manage the standby redo log archive location. A nonempty file specification of the standby redo log archive location for the instance. Specify <code>USE_DB_RECOVERY_FILE_DEST</code> if a database recovery area is desired. A valid <code>LOG_ARCHIVE_DEST_n</code> parameter string with the <code>LOCATION</code> attribute specified, but no <code>VALID_FOR</code>, <code>ALTERNATE</code>, or <code>SERVICE</code> attributes.
Broker default	Empty string
Imported?	No
Parameter class	Dynamic
Role	Primary, standby, and far sync instance

Category	Description
Standby type	Physical, logical, or snapshot standby, or a far sync instance
Corresponds to	<ul style="list-style-type: none"> LOCATION attribute of the LOG_ARCHIVE_DEST_n initialization parameter of the standby database or far sync instance with VALID_FOR compatible with (STANDBY_LOGFILE, ALL_ROLES) DESTINATION column of the V\$ARCHIVE_DEST view of the standby database
Scope	Member
Cloud Control name	Standby Archive Location

 **Note:**

On a logical standby database, Oracle recommends the LOCATION attribute of the LOG_ARCHIVE_DEST_n initialization parameter for the local destination be different from the value of StandbyArchiveLocation property, unless you are using a database recovery area.

StaticConnectIdentifier

The `StaticConnectIdentifier` configurable instance-specific property specifies the connection identifier that the DGMGRL client will use when starting database instances.

If this property has a null value, then the `DGConnectIdentifier` value is used for operations that involve shutting down and starting up the instance.

Category	Description
Datatype	String
Valid values	A connect identifier that refers to a service that is statically registered.
Broker default	Connect descriptor that is the concatenation of: ¹ <ul style="list-style-type: none"> The ADDRESS attribute value of the listener that is specified for the LOCAL_LISTENER initialization parameter The value for the SERVICE_NAME attribute will be set to a concatenation of db_unique_name_DGMGRL.db_domain
Imported?	Yes, from the LOCAL_LISTENER and DB_UNIQUE_NAME initialization parameters.
Parameter class	Not applicable
Role	Primary and standby
Standby type	Physical, logical, or snapshot
Corresponds to	Not applicable
Scope	Instance
Cloud Control name	Not applicable

¹ If the instance specified by the InstanceName property is started with a different SID (read from the INSTANCE_NAME column of the V\$INSTANCE view) than the SID with which it was previously started and/or it

is started on a different host (read from the `HOST_NAME` column of the `V$INSTANCE` view) than the host on which it was previously started, then the broker automatically updates the default value of the `StaticConnectIdentifier` property to incorporate the current `ADDRESS` attribute of the listener that is specified for the `LOCAL_LISTENER` initialization parameter.

See Also:

- See *Oracle Database Net Services Administrator's Guide* for more information about statically registered services
- See the My Oracle Support Note 1387859.1 at <http://support.oracle.com> for additional information about using static services and the `StaticConnectIdentifier` configurable property

TransportDisconnectedThreshold

The `TransportDisconnectedThreshold` configurable property can be used to generate a warning status for a logical, physical, or snapshot standby, or a far sync instance when the last communication from the primary database exceeds the property value.

The property value is expressed in seconds. A value of 0 seconds results in no warnings being generated.

Category	Description
Datatype	Number
Valid values	≥ 0
Broker default	30 seconds
Imported?	No
Parameter class	Not applicable
Role	Standby database, far sync instance
Standby type	Physical, logical, or snapshot standby, or a far sync instance
Corresponds to	Not applicable
Scope	Database, far sync instance
Cloud Control name	Not applicable

TransportLagThreshold

The `TransportLagThreshold` configurable property can be used to generate a warning status for a logical, physical, or snapshot standby, or a far sync instance when the member's transport lag exceeds the property value.

The property value is expressed in seconds. A value of 0 seconds results in no warnings being generated when a transport lag exists.

Category	Description
Datatype	Number

Category	Description
Valid values	>=0
Broker default	30 seconds
Imported?	No
Parameter class	Not applicable
Role	Standby database, far sync instance
Standby type	Physical, logical, or snapshot standby, or a far sync instance
Corresponds to	Not applicable
Scope	Database, far sync instance
Cloud Control name	Not applicable

12

Troubleshooting Oracle Data Guard

Use this information about common issues and resolutions to maintain your Oracle Data Guard environment.

- [Sources of Diagnostic Information](#)
- [General Problems and Solutions](#)
- [Troubleshooting Problems During a Switchover Operation](#)
- [Troubleshooting Problems During a Failover Operation](#)
- [Troubleshooting Problems with the Observer](#)

Sources of Diagnostic Information

The Oracle Data Guard broker provides information about its activities in several forms.

- Database status information (see [Database Status](#))
- Oracle alert log files

The broker records key information in the alert log file for each instance of each database in a broker configuration. You can check the alert log files for such information when troubleshooting Oracle Data Guard.

- Oracle Data Guard "broker log files"

For each instance of each database in a broker configuration, the broker DMON process records important behavior and status information in a broker log file, useful in diagnosing Oracle Data Guard failures. The `TraceLevel` configuration property (see [TraceLevel](#)) is used to specify the level of diagnostic information reported in the broker log files.

The broker log file is created in the same directory as the alert log and is named `drc<${ORACLE_SID}>.log`.

General Problems and Solutions

These topics describe general problems and solutions when using Oracle Data Guard broker.

- [ORA-16596: database not part of the Oracle Data Guard broker configuration](#)
- [Redo Accumulating on the Primary Is Not Sent to Some Standby Databases](#)
- [Many Log Files Are Received on a Standby Database But Not Applied](#)
- [The Request Timed Out or Cloud Control Performance Is Sluggish](#)
- [The Primary Database is Flashed Back](#)
- [Standby Fails to Automatically Start Up Due to Unknown Service \(ORA-12514\)](#)

ORA-16596: database not part of the Oracle Data Guard broker configuration

A request was issued to the broker, but the database instance through which you have connected is no longer a part of the broker configuration.

Solution

Reconnect to the configuration through another database that you know is part of the broker configuration. Confirm that a database exists in the broker configuration that has a name that matches the `db_unique_name` value of the database that returned the ORA-16596 error.

This problem can also occur if you attempt to enable a configuration, but the broker configuration file for one of its databases was accidentally removed or is outdated. In this case, remove the database from the broker configuration, manually delete the configuration file for that standby database (not for the primary database), and try again to enable the configuration. After the configuration is enabled, you can either use the Cloud Control Add Standby Database wizard and choose the Add existing standby database option, or you can use the DGMGRL command-line interface and issue the `ADD DATABASE` command.

Redo Accumulating on the Primary Is Not Sent to Some Standby Databases

By viewing the Log File Details page in Cloud Control, you have determined that log files are accumulating on the primary database and are not being archived to some of the standby databases in the broker configuration.

Solution

To narrow down the problem, do the following:

- Verify that the state of the primary database is in the `TRANSPORT-ON` state (not `TRANSPORT-OFF`).
- Verify that the value of the `LogShipping` database property of the standby database in question is `ON`.
- Check the status of the redo transport services on the primary database using the `LogXptStatus` monitorable property. If redo transport services have an error, then use the error message to determine further checking and resolution action. For example:
 - If the error indicates the standby database is not available, you need to restart the standby database.
 - If the error indicates no listener, you need to restart the listener.
 - If the error indicates the standby database has no local destination, you need to set up a standby location to store archived redo log files from the primary database.

Many Log Files Are Received on a Standby Database But Not Applied

By viewing the Performance page or Log File Details page in Cloud Control, you have determined that the standby database accumulates too many log files without applying them.

Solution

There are many possible reasons why archived redo log files might not be applied to the standby database. Investigate why the log files are building up and rule out the valid reasons.

If the current status of the standby database is not normal:

- Determine whether or not the log apply services might be unexpectedly stopped. See the ORA-16766 (for physical standby databases) or ORA-16768 (for logical standby databases) error description and solution statement for more help.
- If this is a logical standby database, check to see if a failed transaction has occurred.
- If you want to suppress the error while you investigate the problem, you can temporarily disable broker management of the database.

See Also:

[Oracle Data Guard Command-Line Interface Reference](#) for additional information about disabling the database using the DGMGRL command-line interface

If the current status of the standby database is normal:

- Verify the state of the standby database is `APPLY-ON` (not in the `APPLY-OFF` state).
- Verify the state of the primary database is `TRANSPORT-ON` (not in the `TRANSPORT-OFF` state).

See Also:

[Oracle Data Guard Broker Properties](#) for additional information about the `LogShipping` database property

- Check to see if log files are building up because the value of the `DelayMins` property is set too large. (Log apply services will delay applying the archived redo log files on the standby database for the number of minutes specified.)

See Also:

[Oracle Data Guard Broker Properties](#) for additional information about the `DelayMins` database property

- If you cannot see any errors, compare the archive rate to the apply rate on the Performance page in Cloud Control to see if the apply rate is lower than the archive rate.

The Request Timed Out or Cloud Control Performance Is Sluggish

If the broker requests are not completing within the normal timeout parameters, try these actions to solve the problem.

1. Verify the network is operating appropriately.
2. Try to ping all of the nodes in the configuration.
3. Try reconnecting through another database to retry the operation.
4. Run the `VERIFY` command to determine on which database the broker is unable to process the requests.

The Primary Database is Flashed Back

If the primary database is flashed back, then the standby databases in the configuration must be also be flashed back or re-created to be viable targets for switchovers or failovers.

The broker will report errors for the standby databases if the primary database has been flashed back.

For more information about restoring the viability of a standby database that was disabled by the broker, see [Reenabling Disabled Databases After a Role Change](#).

Standby Fails to Automatically Start Up Due to Unknown Service (ORA-12514)

An `ORA-12514` error may be generated if the DGMGRL CLI fails to automatically start up an instance after a broker operation (for example, a switchover, reinstatement, or convert to physical standby).

The full error text is `ORA-12514: TNS:listener does not currently know of service requested in connect descriptor`. If you receive this error, and your database is not managed by Oracle Clusterware, then you must manually start the instance to complete or continue the broker operation.



Note:

The troubleshooting information in this section applicable only to databases that are not managed by Oracle Clusterware.

You can restart the instance before or after completing the following steps:

1. Issue the following DGMGRL CLI command to check the value of the `StaticConnectIdentifier` configurable property for the instance the DGMGRL CLI was unable to restart. (You will have to connect to another running instance to issue this command):

```
SHOW DATABASE db_unique_name StaticConnectIdentifier;
```

2. The static service name specified in the value of the `StaticConnectIdentifier` instance property should be registered with the listener specified in the property value. The default value for the static service name is of the following form:

```
db_unique_name_DGMGRL.db_domain
```

See [Prerequisites](#) for more information about this, and other prerequisites, for using the broker.

3. Confirm that the static service name is registered with the listener specified in the `StaticConnectIdentifier` configurable property value by using the Listener Control utility's status command. If the static service name is properly registered with the listener, it will be included in the output generated by the following command:

```
lsnrctl status
```

Troubleshooting Problems During a Switchover Operation

If the switchover fails due to problems with the configuration, then the broker reports any problems it encounters in the alert log files or in the broker log files.

See [Sources of Diagnostic Information](#) for more information about log files. If the reported problems can be corrected, you can retry the switchover operation and it will usually succeed. If the reported problems cannot be corrected or the switchover operation fails even after correcting the reported problems, then you can choose another database for the switchover or restore the configuration to its pre-switchover state and then retry the switchover.

If fast-start failover is enabled, the broker does not allow switchover to any standby database except to the target standby database. In addition, switchover to the target standby database is allowed only when the value of the `FS_FAILOVER_STATUS` column in the `V$DATABASE` view on the target standby database is either `READY` or `SUSPENDED`.



See Also:

[When Fast-Start Failover Is Enabled and the Observer Is Running](#)

Troubleshooting Problems During a Failover Operation

Although it is possible for a failover operation to fail, it is unlikely. If an error does occur there are guidelines you can follow to fix the problem and then retry the broker failover.

Failed Failovers to Physical Standby Databases

These steps describe how to recover from a failed broker failover to a physical standby database.

Failed Broker Complete Physical Failovers

Examine the alert log file and the broker log file (drc*.log) on the target standby database to determine the cause of the failure and correct the problem.

If the reported problem can be corrected, then retry the failover operation. If the reported problem cannot be corrected or if the failover operation fails again after the reported problem has been corrected, then take the following steps:

1. Connect to the target standby database and disable fast-start failover using the `FORCE` option if it is enabled.
2. Then you can either:
 - Connect to another physical standby database and attempt a broker complete failover.
 - Perform a broker immediate failover to the target physical standby database.
3. Reinstall the original primary database and any bystander physical standby databases that are disabled with a status of reinstatement required (ORA-16661).
4. Reenable fast-start failover if it was disabled in step 1.

Failed Broker Immediate Physical Failovers

Examine the alert log file and the broker log file (drc*.log) on the target standby database to determine the cause of the failure and correct the problem.

If the problem can be corrected, retry the broker immediate failover. Otherwise connect to another physical standby database and attempt either a broker complete or immediate failover.

Failed Failovers to Logical Standby Databases

These are the steps to follow if a failover to a logical standby database fails.

1. Examine the alert log file and the broker log file (drc*.log) on the target standby database to determine the cause of the failure and correct the problem.
2. Connect to the target standby database and disable fast-start failover using the `FORCE` option if it is enabled.
3. Retry the broker failover.
4. Reinstall the old primary database. All bystander standby databases will be re-created from a copy of the new primary database.
5. Reenable fast-start failover if it was disabled in step 1.

If broker failover continues to fail, you should stop the broker on all databases in the Oracle Data Guard configuration (set the `DG_BROKER_START` initialization parameter to `FALSE`). Remove the Oracle Data Guard broker configuration files from all databases. Attempt a manual failover using the guidelines for role transitions in *Oracle Data Guard Concepts and Administration*.

**Note:**

You can enable or disable the broker configuration using `DGMGRL ENABLE CONFIGURATION` and `DISABLE CONFIGURATION` commands. You cannot disable the configuration using Cloud Control. You can only enable the configuration using Cloud Control if it was previously disabled using `DGMGRL`.

Troubleshooting Problems with the Observer

The observer continuously monitors the fast-start failover environment to ensure the primary database is available.

Installing and starting the observer is an integral part of using fast-start failover. The following sections describe techniques for troubleshooting the observer:

- [Problems Because the Observer Has Stopped](#)
- [Capturing Observer Actions in the Observer Log File](#)

Problems Because the Observer Has Stopped

If the observer host machine crashes, the broker configuration is no longer observed and fast-start failover is no longer possible.

In this case, you may have to move the observer to a new host if the original host machine cannot be repaired in a timely fashion.

1. Issue the `DGMGRL STOP OBSERVER` command to sever the link between the original observer and the broker configuration:

```
DGMGRL> STOP OBSERVER;
Done.
```

2. Issue the `DGMGRL SHOW CONFIGURATION VERBOSE` command to verify that the configuration is no longer being observed:

```
DGMGRL> SHOW CONFIGURATION VERBOSE;
```

```
Configuration - DRSolution
```

```
Protection Mode: MaxAvailability
```

```
Members:
```

```
North_Sales - Primary database
```

```
Warning: ORA-16819: fast-start failover observer not started
```

```
South_Sales - (*) Physical standby database
```

```
Warning: ORA-16819: fast-start failover observer not started
```

```
(*) Fast-Start Failover target
```

```
Properties:
```

```
FastStartFailoverThreshold = '30'
```

```
OperationTimeout = '30'
```

```
TraceLevel = 'USER'
```

```
FastStartFailoverLagLimit = '30'
```

```
CommunicationTimeout = '180'
```

```

ObserverReconnect           = '0'
FastStartFailoverAutoReinstate = 'TRUE'
FastStartFailoverPmyShutdown = 'TRUE'
BystandersFollowRoleChange   = 'ALL'
ObserverOverride            = 'FALSE'
ExternalDestination1         = ''
ExternalDestination2         = ''
PrimaryLostWriteAction       = 'CONTINUE'
ConfigurationWideService     = 'North_Sales_CFG'

```

Fast-Start Failover: ENABLED

```

Threshold: 30 seconds
Target: South_Sales
Observer: (none)
Lag Limit: 30 seconds (not in use)
Shutdown Primary: TRUE
Auto-reinstate: TRUE
Observer Reconnect: (none)
Observer Override: FALSE

```

Configuration Status:
WARNING

3. Note that you do not need to issue the `DGMGRL SHOW CONFIGURATION` command to verify that the observer has actually stopped. Successful completion of the `DGMGRL STOP OBSERVER` command will allow a new observer to become associated with the configuration.

Capturing Observer Actions in the Observer Log File

You can use the `LOGFILE IS` option of the `START OBSERVER` command to capture the activity performed by the observer.

Start DGMGRL and issue the command:

```
DGMGRL> START OBSERVER observer1 IN BACKGROUND LOGFILE IS observer.log
CONNECT IDENTIFIER IS North_Sales TRACE_LEVEL IS SUPPORT;
```

When the `LOGFILE IS` clause is used, then all observer output is recorded in the specified file. If a complete path, with file name, is provided, the file is stored in the specified path. If only a file name is provided and the `DG_ADMIN` environment variable is defined, the specified file is stored in the `$DG_ADMIN/config_ConfigurationSimpleName/log` directory. If the `DG_ADMIN` environment variable is not defined, the file is stored in the current working directory. If `LOGFILE IS` clause is omitted, the log file is stored in the `$DG_ADMIN/config_ConfigurationSimpleName/log` directory using the name `observer_hostname.log`. If the `DG_ADMIN` environment variable is not defined, the log file is stored as `observer_hostname.log` in the current working directory. `ConfigurationSimpleName` is the name of the broker configuration.

A

Oracle Data Guard Broker Upgrading and Downgrading

Use these topics to upgrade or downgrade Oracle databases and Oracle Enterprise Manager Cloud Control (Cloud Control) in a broker configuration.



See Also:

[Oracle Data Guard Installation](#)

Considerations While Using the DBMS_ROLLING Package

When upgrading from Oracle Database Version 12.2 to a higher database version, broker management can remain enabled. However, some considerations must be kept in mind.

The considerations are as follows:

- The broker configuration files must be in a release-neutral location and accessible throughout the upgrade process.
- After upgrading the leading group members, ensure broker communication between the leading group and the trailing group is successful before proceeding with the upgrade process.

Communication may be disrupted if network files are either not moved to the upgraded `ORACLE_HOME` or need to be edited appropriately. In such cases, shut down the upgraded databases (transient logical standby and any other Leading Group Standby databases), stop the listener, copy the `listener.ora` and `tnsnames.ora` or fix them as appropriate, restart the listener, and restart the databases.

- Before you run the `DBMS_ROLLING` procedure, the physical standby databases must have been operating in Active Data Guard mode.

If the physical standby databases were operating only in the mounted mode, and the primary database is a multitenant database, you must open the primary database including all its PDBs, and the temp file configuration must be confirmed before you run procedures in the `DBMS_ROLLING` package. This ensures that the necessary temp files are created and available during the upgrade process.

Glossary

apply instance

In an Oracle Real Application Clusters (Oracle RAC) databases environment, the apply instance is the one instance applying archived redo data to a standby database.

broker

A distributed management framework that automates and simplifies most of the complex operations required to create, control, and monitor an Oracle Data Guard configuration.

broker configuration

A logical grouping of the primary and standby databases (including redo transport services and log apply services) of an Oracle Data Guard configuration.

See also [Data Guard configuration](#).

bystander standby database

A standby database that is not the target of, or directly involved in, a switchover or failover operation.

configuration object

A named collection of database objects. It is an abstraction of an Oracle Data Guard configuration.

database guard

The database guard controls whether or not modifications can be made to the tables in a logical standby database.

database object

A named object that corresponds to a primary or standby database in an Oracle Data Guard configuration. The broker uses this object to manage and control the state of a single database and to associate properties with the database.

Data Guard command-line interface

The Oracle Data Guard command-line interface (DGMGRL) enables you to control and monitor an Oracle Data Guard configuration from the DGMGRL prompt or within scripts.

Data Guard configuration

A distributed computing system that prevents or minimizes losses due to unplanned events (for example, human errors, environmental disasters, or data corruption) as well as to planned downtime (such as for routine maintenance tasks).

See *also* [broker configuration](#).

Data Guard environment

The physical configuration of the primary, standby databases, and far sync instances. The environment depends on many factors, including the:

- Number of standby databases and far sync instances associated with a primary database
- Number of host systems used by the databases
- Directory structures of the machines used by the databases
- Network configuration
- Redo transport services
- Log apply services

The Oracle Data Guard environment can be managed manually by a DBA, automatically using Enterprise Manager or the Data Guard command-line interface (DGMGRL) or a combination of all of these.

default state

The initial runtime state in which the database object will run when you enable broker management of the configuration. The actual default state can vary depending on the role (primary or standby) in which the database is running.

See *also* [intended state](#).

failover

Failover changes a standby database into the role of a primary database and disables the old primary database.

See *also* [fast-start failover](#) and [manual failover](#).

far sync instance

A remote Oracle Data Guard destination that accepts redo from the primary database and redistributes that redo throughout the Oracle Data Guard configuration. It is similar

to a physical standby database in that it manages a control file, receives redo into Standby Redo Logs (SRLs), and archives those SRLs to local Archived Redo Logs (ARLs). But unlike a standby database, a far sync instance does not manage data files, cannot be opened for access, and cannot run redo apply.

fast-start failover

Enables a failover *automatically* when the primary database becomes unavailable. When fast-start failover is enabled, the broker determines if a failover is necessary and automatically, quickly, and reliably fails over to a designated standby without requiring that you perform any manual steps to invoke the failover.

See *also* [manual failover](#).

instance object

A named object; a database object is a collection of one or more named instance objects. The broker uses this object to manage and control the state of the database with which the instance is associated, and to associate instance-specific properties with this instance of the database.

intended state

The runtime state of a database object while it is enabled for management by the broker.

See *also* [default state](#).

lights out computing

Refers to computing equipment or software whose operations are automated, requiring little to no intervention by human administrators.

The term "lights out" originated from when computing centers were located in one room and contained a number of servers that were kept under lock and key and in the dark. Under normal operation, the room was not entered by human administrators, and all operations in the room were automated.

logical standby database

A logical standby database takes standard Oracle archived redo log files, transforms them back into SQL transactions, and then applies them to an open standby database. Although changes can be applied concurrently with end-user access, the tables being maintained through regenerated SQL transactions allow read-only access to users of the logical standby database. Because the database is open to allow application of reconstructed SQL transactions, it is physically different from the primary database. The database tables can have different indexes and physical characteristics from their primary database peers, but the

tables must maintain logical consistency from an application access perspective, to fulfill their role as a standby data source.

manual failover

A failover that is initiated by the DBA who first determines a failover is necessary and then invokes one by clicking `FAILOVER` in Enterprise Manager or by issuing the `DGMGRL FAILOVER` command. The word *manual* is used to contrast this type of failover with a fast-start failover.

See also [fast-start failover](#).

observer

A DGMGRL client that continuously monitors the primary and target standby databases, evaluates whether failover is necessary, and initiates a fast-start failover when conditions warrant.

physical standby database

A physical standby database is an exact copy of a primary database. While the primary database is open and active, a physical standby database uses Redo Apply to apply redo data received from the primary database. Redo Apply can run on a physical standby database instance that is mounted. If a license for the Oracle Active Data Guard option has been purchased, Redo Apply can also run on a physical standby database instance that is open. See *Oracle Data Guard Concepts and Administration* for more details.

primary database

A production database from which one or more standby databases is created and maintained. Every standby database is associated with one and only one primary database. A single primary database can, however, support multiple standby databases. The Oracle Data Guard broker monitor (DMON) maintains the master copy of the binary configuration file with the primary database, ensuring that each standby database's copy of the file is kept up to date as changes are made.

The broker refers to this database using the value in the `DB_UNIQUE_NAME` initialization parameter which is defined to be globally unique.

read-only mode

A mode in which a database can be opened that allows queries, but disallows modifications.

A physical standby database can be opened read-only so that queries may be performed. If a license for the Oracle Active Data Guard option has been purchased, a physical standby database can be open while Redo Apply is active. This capability is

known as real-time query. See *Oracle Data Guard Concepts and Administration* for more details.

Redo Apply

A physical standby database is kept synchronized with the primary database through Redo Apply, which recovers the redo data received from the primary database and applies the redo to the physical standby database.

reinstatement

Reinstatement is the process of turning a database, including the old primary database, that had been disabled after a failover operation into a viable standby database for the new primary database. Flashback database must be enabled on a database in order to reinstate it.

snapshot standby database

A snapshot standby is a fully updatable standby database that is created from a physical standby database. On snapshot standby databases, redo data is received but not applied until the snapshot standby database is converted back to a physical standby database.

SQL Apply

A logical standby database is kept synchronized with the primary database through SQL Apply, which transforms the data in the redo received from the primary database into SQL statements and then executes the SQL statements on the standby database.

standby database

A copy of a primary database created using a backup of your primary database. Standby databases are kept synchronized with the primary database by applying archived redo data over time from the primary database to each standby database. The standby database can take over processing from the primary database, providing nearly continuous database availability. A standby database has its own server parameter file, control file, and data files. It also has a copy of the broker's configuration file, kept up to date at the direction of the DMON process running in the primary database instance.

The broker refers to a standby database by its globally unique name that is stored in its `DB_UNIQUE_NAME` initialization parameter.

See *also* [logical standby database](#), [physical standby database](#) and [snapshot standby database](#).

Index

A

ADD CONFIGURATION, [10-11](#)
ADD DATABASE command, [10-12](#)
ADD FAR_SYNC command, [10-13](#)
ADD PLUGGABLE DATABASE, [10-13](#)
Add Standby Database wizard
 creating a broker configuration, [3-11](#)
 definition, [1-10](#)
 introduction, [1-10](#)
altering
 properties
 database, [8-15](#)
 states
 database, [8-16](#)
AlternateLocation property, [11-24](#)
application continuity
 support in Oracle Data Guard broker, [6-68](#)
application integration, [1-6](#)
apply instance, [4-21](#)
 failover, [4-21](#)
 PreferredApplyInstance property, [4-21](#)
 selecting, [4-21](#)
ApplyInstanceTimeout property, [11-27](#)
ApplyLagThreshold property, [11-27](#)
ApplyParallel property, [11-28](#)
architecture
 Data Guard broker, [1-9](#)
archived redo logs
 destination and configuration parameters, [3-1](#)
 in a Data Guard configuration, [3-1](#)
 primary database setup, [2-2](#)
ArchiveLocation property, [11-25](#)
ASYNCRedo transport service, [4-9](#)

B

background processes
 DMON, [1-12](#)
banners
 suppressing from DGMGRL command-line
 interface output, [10-1](#)
batch programs
 using Data Guard command-line interface
 (DGMGRL), [10-1](#)

before you perform a switchover, [6-5](#)
before you use DGMGRL, [8-1](#)
benefits
 Data Guard broker, [1-6](#)
 high availability, [1-6](#)
 scalability, [1-6](#)
 with Oracle Real Application Clusters, [1-6](#)
Binding property, [11-28](#)
broker
 components, [1-9](#)
 configuration, [1-3](#)
 Data Guard configuration file, [1-14](#)
 drc* log files, [12-1](#)
 introduction, [1-3](#)
 user interfaces, [1-10](#)
broker configuration
 exporting, [8-34](#)
 importing, [8-34](#)
broker configurations
 benefits, [1-6](#)
 changing roles, [3-11](#)
 creating, [8-2](#)
 Data Guard configuration file, [1-12](#)
 disabling databases, [3-14](#), [10-22](#)
 effects of removing metadata from, [10-64](#)
 enabling, [8-7](#), [10-42](#)
 enabling databases, [3-11](#), [3-14](#), [8-7](#), [10-42](#)
 enabling fast-start failover, [10-44](#)
 health, [1-12](#), [3-15](#)
 life cycle, [3-11](#)
 management, [1-10](#), [1-11](#), [1-14](#)
 monitoring, [8-31](#)
 Oracle Net Services configuration, [1-6](#), [1-11](#)
 properties, [4-5](#)
 protection modes, [10-26](#)
 setting protection mode, [4-24](#)
 status modes, [3-15](#)
 supported, [3-1](#)
broker processes
 monitoring, [1-12](#)
bystander standby databases, [6-4](#)
 reintegrating into configuration, [6-12](#)
BystandersFollowRoleChange property, [11-3](#)

C

CDBs

Data Guard broker support, [1-5](#)

CFS

See cluster file system

changing, [10-26](#)

properties

databases in a broker configuration,
[3-11](#), [4-7](#)

roles

within the broker configuration, [3-11](#)

states

databases in a broker configuration, [4-3](#)
of databases in a broker configuration,
[8-15](#)

cluster file system (CFS)

broker configuration files, [3-9](#)

command prompts

suppressing from DGMGRL command-line
interface output, [10-1](#)

command-line interface

See Data Guard command-line interface
(DGMGRL)

CommunicationTimeout property, [11-4](#)

complete failover, [6-9](#), [10-49](#)

components

broker, [1-9](#)

Data Guard configuration, [3-1](#)

configurable properties, [4-5](#)

AlternateLocation, [11-24](#)

ApplyInstanceTimeout, [11-27](#)

ApplyLagThreshold, [11-27](#)

ApplyParallel, [11-28](#)

ArchiveLocation, [11-25](#)

Binding, [11-28](#)

database, [4-7](#), [11-22](#)

DelayMins, [11-29](#)

DGConnectIdentifier, [11-30](#)

FastStartFailoverTarget, [11-32](#)

InstanceName, [11-33](#)

LogShipping, [11-33](#)

LogXptMode, [11-34](#)

managing redo transport services, [4-8](#)

MaxFailure, [11-35](#)

NetTimeout, [11-36](#)

ObserverConnectIdentifier, [11-37](#)

PreferredApplyInstance, [11-37](#)

RedoCompression, [11-39](#)

RedoRoutes, [11-39](#)

ReopenSecs, [11-43](#)

resetting to default values, [4-8](#)

StandbyAlternateLocation, [11-43](#)

StandbyArchiveLocation, [11-44](#)

StaticConnectIdentifier, [11-45](#)

configurable properties (*continued*)

TransportDisconnectedThreshold, [11-46](#)

TransportLagThreshold, [11-46](#)

configuration files

fast-start failover configuration (fsfo.dat) file,
[6-57](#)

See Also Data Guard configuration file, [1-14](#)

configuration properties, [3-7](#)

BystandersFollowRoleChange, [11-3](#)

CommunicationTimeout, [11-4](#)

FastStartFailoverAutoReinstate, [11-8](#)

FastStartFailoverLagLimit, [11-8](#)

FastStartFailoverPmyShutdown, [11-9](#)

FastStartFailoverThreshold, [11-9](#)

ObserverOverride, [11-10](#)

ObserverPingInterval, [11-11](#)

ObserverPingRetry, [11-11](#)

ObserverReconnect, [11-12](#)

OperationTimeout, [11-12](#)

PrimaryLostWriteAction, [11-14](#)

Tracelevel, [11-15](#)

connecting

privileges required for Data Guard broker
configurations, [10-7](#)

starting the observer, [8-12](#)

CONVERT DATABASE command, [10-17](#)

CREATE CONFIGURATION command, [10-19](#)

CREATE FAR_SYNC command, [10-20](#)

creating

a broker configuration, [8-2](#)

with the Add Standby Database wizard,
[3-11](#)

a standby database, [3-11](#)

D

Data Guard

troubleshooting, [12-1](#)

Data Guard broker

See Also broker, [1-3](#)

Data Guard command line interface

echoing commands, [10-69](#), [10-75](#)

Data Guard command-line interface

commands

ENABLE FAST_START FAILOVER,
[10-44](#)

SET STATE, [4-21](#)

DG_BROKER_START initialization

parameter, [2-2](#)

enabling a database, [8-7](#)

introduction, [1-11](#)

Data Guard command-line interface (DGMGRL),

[8-1](#)

banners from output, [10-1](#)

- Data Guard command-line interface (DGMGRL) (*continued*)
- commands
 - ADD DATABASE, [10-12](#)
 - ADD FAR_SYNC, [10-13](#)
 - CONVERT DATABASE, [10-17](#)
 - CREATE CONFIGURATION, [10-19](#)
 - CREATE FAR_SYNC, [10-20](#)
 - DISABLE CONFIGURATION, [10-22](#)
 - DISABLE DATABASE, [10-23](#)
 - DISABLE FAR_SYNC, [10-19](#), [10-23](#)
 - DISABLE FAST_START FAILOVER, [10-24](#)
 - DISABLE FAST_START FAILOVER CONDITION, [10-25](#)
 - EDIT CONFIGURATION (property), [10-26](#)
 - EDIT CONFIGURATION (protection mode), [10-26](#)
 - EDIT CONFIGURATION (RENAME), [10-19](#), [10-28](#)
 - EDIT CONFIGURATION RESET (Property), [10-19](#), [10-29](#)
 - EDIT DATABASE (Property), [10-19](#), [10-30](#)
 - EDIT DATABASE (rename), [10-33](#)
 - EDIT DATABASE (state), [10-34](#)
 - EDIT DATABASE RESET (Parameter), [10-35](#)
 - EDIT DATABASE RESET (Property), [10-34](#)
 - EDIT FAR_SYNC, [10-35](#)
 - EDIT FAR_SYNC RESET (Parameter), [10-37](#)
 - EDIT FAR_SYNC RESET (Property), [10-37](#)
 - ENABLE CONFIGURATION, [10-42](#)
 - ENABLE DATABASE, [10-43](#)
 - ENABLE FAR_SYNC, [10-44](#)
 - ENABLE FAST_START FAILOVER, [10-44](#)
 - ENABLE FAST_START FAILOVER CONDITION, [10-46](#)
 - EXIT, [10-48](#)
 - EXPORT CONFIGURATION, [10-49](#)
 - FAILOVER, [8-26](#)
 - HELP, [10-53](#)
 - IMPORT CONFIGURATION, [10-55](#)
 - PREPARE DATABASE FOR DATA GUARD, [10-60](#)
 - QUIT, [10-63](#)
 - REINSTATE DATABASE, [10-63](#)
 - REMOVE CONFIGURATION, [10-64](#)
 - REMOVE DATABASE, [10-65](#)
 - REMOVE FAR_SYNC, [10-66](#)
 - REMOVE INSTANCE, [10-67](#)
- Data Guard command-line interface (DGMGRL) (*continued*)
- commands (*continued*)
 - SET FAST_START FAILOVER TARGET, [10-70](#)
 - SHOW CONFIGURATION, [10-76](#), [10-79](#)
 - SHOW DATABASE, [10-81](#)
 - SHOW FAR_SYNC, [10-85](#)
 - SHOW FAST_START FAILOVER, [10-86](#)
 - SHOW INSTANCE, [10-88](#)
 - SHUTDOWN, [10-95](#)
 - SQL, [10-97](#)
 - START OBSERVER, [10-97](#)
 - STARTUP, [10-103](#)
 - STOP OBSERVER, [10-105](#)
 - SWITCHOVER, [8-21](#), [10-107](#)
 - VALIDATE DATABASE, [10-113](#)
 - VALIDATE FAR_SYNC, [10-122](#)
 - creating a configuration, [8-2](#)
 - DG_BROKER_START initialization parameter, [10-7](#)
 - enabling the configuration, [8-7](#)
 - monitoring broker configurations, [8-31](#)
 - performing routine management tasks, [8-15](#)
 - prerequisites, [8-1](#)
 - setting critical database properties, [8-5](#)
 - setting up standby redo log files, [4-26](#)
 - single command mode, [10-1](#)
 - starting, [10-1](#)
 - stopping, [10-9](#)
 - string values, [10-7](#)
 - summary of commands, [10-3](#)
 - suppressing command prompts, [10-1](#)
- Data Guard configuration file
- for an Oracle RAC database, [3-9](#)
 - in a CFS area, [3-9](#)
 - inconsistent values from server parameter file, [11-17](#)
 - relationship to DMON process, [1-12](#)
 - setting up, [3-7](#)
- Data Guard configurations
- automated creation of, [1-6](#)
 - background, [3-1](#)
- Data Guard monitor (DMON), [1-11](#)
- interaction with the Oracle database, [1-12](#)
 - managing objects, [3-1](#), [3-11](#), [8-7](#)
 - Oracle database background processes, [1-12](#)
 - overview, [1-12](#)
 - two-way network communication, [1-12](#)
- data protection modes
- See protection modes
- database resources
- monitoring, [1-6](#)
- database status
- querying, [4-37](#)

databases

- changing the state of, [8-15](#)
- creating and adding to a broker configuration, [1-10](#)
- dependencies, [4-1](#)
- disabling management of, [10-22](#)
- during
 - fast-start failover, [6-17](#)
 - manual failover, [6-8](#)
 - switchover, [6-4](#)
- enabling, [8-7](#)
- health, [4-37](#)
- installation for broker management, [2-2](#)
- monitorable properties, [4-7](#)
- network setup, [1-12](#), [2-2](#)
- properties, [1-11](#), [4-7](#)
- property management, [1-14](#)
- reenabling after a role transition, [6-15](#)
- restarting
 - after fast-start failover, [6-63](#)
- shutting down
 - effect on fast-start failover, [6-34](#), [6-38](#)
- states, [4-1](#)
 - dependencies, [4-1](#)
 - transitions, [4-3](#)
- status, [4-37](#)

DBMS_DG package

- using to direct fast-start failover, [6-38](#)

delaying

- application of redo data with DelayMins property, [4-19](#)

DelayMins property, [11-29](#)

destinations

- archived redo log file parameters, [3-1](#)
- viewing the LogXptStatus property, [11-18](#)

DG_BROKER_CONFIG_FILEn file, [3-9](#)

- in a CFS area, [3-9](#)

DG_BROKER_START initialization parameter, [2-2](#), [8-1](#), [10-7](#)

DGConnectIdentifier property, [11-30](#)

DGMGRL

- reinstating a database, [6-15](#)

diagnostic information

- sources, [12-1](#)

DISABLE CONFIGURATION command, [10-22](#)

- example, [8-17](#)

DISABLE DATABASE command, [10-23](#)

- example, [8-17](#)

DISABLE FAR_SYNC command, [10-19](#), [10-23](#)

DISABLE FAST_START FAILOVER command, [10-24](#)

DISABLE FAST_START FAILOVER CONDITION command, [10-25](#)

disabling, [10-22](#)

- broker configuration, [8-16](#), [10-22](#)

disabling (*continued*)

- broker management of standby database, [10-23](#)
- databases, [8-17](#)
- fast-start failover, [6-45](#), [10-24](#)
 - on a standby database, [6-45](#)

disaster protection

- benefits, [1-6](#)

displaying

- configuration information, [10-76](#)
- help for DGMGRL commands, [10-53](#)

distributed management framework, [1-3](#)

DMON

- See Data Guard monitor (DMON)

downgrading

- protection mode, [4-28](#)

drc* log files

- broker diagnostic information, [12-1](#)
- during reinstatement, [6-42](#)
- recording failed reinstatement, [6-63](#)

E

e-mail

- reporting events, [1-11](#)

EDIT CONFIGURATION (Property) command, [10-26](#)

EDIT CONFIGURATION (protection mode) command, [10-26](#)

EDIT CONFIGURATION (RENAME) command, [10-19](#), [10-28](#)

EDIT CONFIGURATION (Set Source) command, [10-29](#)

EDIT CONFIGURATION RESET (Property) command, [10-19](#), [10-29](#)

EDIT DATABASE (Parameter) command, [10-32](#)

EDIT DATABASE (property) command

- example, [8-15](#)

EDIT DATABASE (Property) command, [10-19](#), [10-30](#)

EDIT DATABASE (rename) command, [10-33](#)

EDIT DATABASE (state) command, [10-34](#)

- example, [8-16](#)

EDIT DATABASE RESET (Parameter) command, [10-35](#)

EDIT DATABASE RESET (Property) command, [10-34](#)

EDIT FAR_SYNC command, [10-35](#)

EDIT FAR_SYNC RESET (Parameter) command, [10-37](#)

EDIT FAR_SYNC RESET (Property) command, [10-37](#)

EDIT PLUGGABLE DATABASE command, [10-38](#)

EDIT RECOVERY_APPLIANCE (Parameter) command, [10-39](#)

- editing
 - configurable properties, [10-26](#)
 - database name, [10-33](#)
 - database state, [10-34](#)
 - protection modes, [10-26](#)
 - ENABLE CONFIGURATION command, [8-7](#), [10-42](#)
 - ENABLE DATABASE command, [10-43](#)
 - ENABLE FAR_SYNC command, [10-44](#)
 - ENABLE FAST_START FAILOVER command, [10-44](#)
 - ENABLE FAST_START FAILOVER CONDITION command, [10-46](#)
 - enabling, [10-44](#)
 - broker configuration, [3-11](#), [8-7](#)
 - Enterprise Edition database
 - installation, [2-1](#)
 - Enterprise Manager
 - Fast-Start Failover wizard
 - disabling fast-start failover, [6-45](#)
 - wizards
 - automate standby database creation, [1-4](#)
 - error messages
 - ORA-16795, [6-15](#)
 - error status, [3-15](#)
 - events
 - monitoring with Oracle Enterprise Manager, [1-4](#)
 - Oracle Enterprise Manager, [1-6](#)
 - reporting, [1-11](#)
 - responding to, [1-6](#)
 - EXIT command, [10-48](#), [10-63](#)
 - See also QUIT command
 - EXPORT CONFIGURATION command, [10-49](#)
 - exporting
 - broker configuration, [8-33](#), [8-34](#)
- ## F
-
- failover
 - failing over to a standby database, [10-49](#)
 - Oracle Enterprise Manager, [1-11](#)
 - FAILOVER command, [8-26](#), [10-49](#)
 - failovers
 - benefits, [1-6](#)
 - broker tasks, [6-12](#)
 - choosing a target standby database, [6-2](#)
 - complete, [6-12](#)
 - fast-start failover, [6-36](#)
 - immediate, [6-14](#)
 - managing, [6-1](#)
 - fast-start, [6-17](#)
 - manual, [6-8](#)
 - manual
 - complete option, [6-9](#)
 - failovers (*continued*)
 - manual (*continued*)
 - effect on data protection mode, [6-11](#)
 - immediate option, [6-8](#)
 - reenabling disabled databases after, [6-12](#)
 - starting manually, [6-10](#)
 - using DGMGRL, [8-26](#)
 - failures
 - observer, [6-54](#)
 - far sync instances
 - managing, [4-31](#)
 - fast-start failover
 - configuration information in the fsfo.dat file, [6-57](#)
 - DGMGRL client as the observer, [1-11](#)
 - directed from an application, [6-38](#)
 - disabling, [6-45](#)
 - to perform manual failover, [6-47](#)
 - enabling, [10-44](#)
 - instance failures, [6-34](#)
 - monitoring by the observer, [6-34](#)
 - observer overview, [6-17](#)
 - performing manual failovers when enabled, [6-38](#)
 - performing switchover when enabled, [6-38](#)
 - querying V\$DATABASE, [6-36](#)
 - reinstating the former primary database, [6-62](#)
 - restarting databases after shut down, [6-63](#)
 - setting the FastStartFailoverThreshold property, [6-26](#)
 - shutdown abort, [6-34](#), [6-38](#)
 - to Oracle RAC database instances, [6-35](#)
 - tuning, [4-33](#)
 - unobserved, [6-54](#)
 - viewing failover configuration statistics, [6-41](#)
 - FastStartFailoverAutoReinstate property, [11-8](#)
 - FastStartFailoverLagLimit property, [11-8](#)
 - FastStartFailoverPmyShutdown property, [11-9](#)
 - FastStartFailoverTarget property, [11-32](#)
 - FastStartFailoverThreshold property, [11-9](#)
 - setting, [6-26](#)
 - FASTSYNC redo transport mode, [11-34](#)
 - FASTSYNC redo transport service, [4-9](#)
 - files
 - naming the server parameter file, [1-14](#)
 - Flashback Database
 - reinstating databases, [6-15](#)
 - use after failover, [6-12](#)
 - flashback logs
 - reinstating a failed primary database, [6-15](#)
 - FORCE option
 - performing a manual failover, [6-47](#)
 - FS_FAILOVER_OBSERVER_HOST column
 - of V\$DATABASE view, [6-52](#)

FS_FAILOVER_OBSERVER_PRESENT column
of V\$DATABASE view, [6-52](#)

FS_FAILOVER_STATUS column
of V\$DATABASE view
failed reinstatement, [6-42](#)

FSFO.DAT file
overview, [6-57](#)

H

health check, [1-6](#)
monitoring, [1-12](#)
on primary database, [4-37](#)
on standby database, [4-37](#)
revealed by configuration status, [3-15](#)

HELP command, [10-53](#)

high availability
benefits, [1-6](#)
levels of data protection, [4-24](#), [11-34](#)
LogXptMode property, [11-34](#)
restoring after fast-start failover, [6-62](#)

I

immediate failover, [6-8](#), [10-49](#)

IMPORT CONFIGURATION command, [10-55](#)

importing
broker configuration, [8-33](#), [8-34](#)

InconsistentLogXptProps property, [11-16](#)

InconsistentProperties property, [11-17](#)

increased scalability
benefits, [1-6](#)

initialization parameters
database configurable properties, [4-7](#)
DG_BROKER_CONFIG_FILEn, [3-9](#)
DG_BROKER_START, [2-2](#), [8-1](#)
dynamic, [4-7](#)
static, [4-7](#)

installation
ARCHIVELOG mode setup, [2-2](#)
Data Guard, [2-1](#)
Oracle Enterprise Manager, [2-1](#)
Oracle Instant Client, [2-1](#), [10-97](#)
prerequisites, [2-2](#)

InstanceName property, [11-33](#)

instances
failures, [6-34](#)
removing, [10-25](#), [10-46](#)
shutting down, [10-95](#)
starting, [10-103](#)

intended state
configuration health check, [3-15](#)

L

life cycle of a broker configuration, [3-11](#)

log apply services
and apply instance failover, [4-21](#)
and parallel apply, [4-20](#)
apply instance, [4-21](#)
configuring, [1-11](#)
Data Guard configuration, [3-1](#)
delayed application of redo data, [4-19](#)
delaying, [11-29](#)
in an Oracle RAC database, [4-21](#)
managing, [4-19](#)
selecting the apply instance, [4-21](#)
verifying, [1-6](#)

log files
broker, [6-63](#)

logical standby databases
state transitions, [4-3](#)
switchover to the primary role, [6-4](#)

LogShipping property, [11-33](#)

LogXptMode property, [11-34](#)

LogXptStatus property, [11-18](#)

LsbyFailedTxnInfo property, [11-18](#)

LsbyParameters property, [11-19](#)

M

management
benefits, [1-6](#)
Oracle Enterprise Manager, [1-10](#)

managing
a broker configuration, [8-1](#)
broker configurations, [3-1](#)
databases, [4-1](#)
fast-start failover, [6-17](#)
local operations, [1-3](#)
log apply services, [4-19](#)
manual failover, [6-8](#)
parallel apply in a physical standby database,
[4-20](#)
remote operations, [1-3](#)
roles, [6-1](#)
fast-start failover, [6-17](#)
manual failover, [6-8](#)
switchover, [6-4](#)
switchover, [6-4](#)

manual failover, [1-8](#), [6-1](#)
complete option, [6-9](#)
effect on protection modes, [4-29](#)
immediate option, [6-8](#)
performing when fast-start failover is
enabled, [6-38](#), [6-47](#)
starting, [6-10](#)

manual role changes
 performing when fast-start failover is enabled, [6-38](#)

MaxFailure property, [11-35](#)

maximize availability, [1-6](#), [11-34](#)

maximize data protection, [1-6](#), [11-34](#)

maximize performance, [1-6](#), [11-34](#)

maximum availability
 data protection mode, [4-25](#)

maximum performance
 data protection mode, [4-25](#)

maximum protection
 data protection mode, [4-25](#)

monitorable properties, [4-5](#), [4-7](#), [11-16](#)
 InconsistentLogXptProps, [11-16](#)
 InconsistentProperties, [11-17](#)
 LogXptStatus, [11-18](#)
 LsbyFailedTxnInfo, [11-18](#)
 LsbyParameters, [11-19](#)
 RecvQEntries, [11-19](#)
 SendQEntries, [11-20](#)
 TopWaitEvents, [11-22](#)

monitoring
 broker configurations, [1-12](#), [8-1](#), [8-31](#)
 broker processes, [1-12](#)
 local and remote databases, [1-6](#)
 Oracle Enterprise Manager performance page, [3-11](#)
 through the command-line interface, [1-11](#)

multitenant container databases
 See CDBs, [1-5](#)

N

NetTimeout property, [11-36](#)

network configurations
 validating before role changes, [4-40](#)

networks
 Data Guard configuration, [3-1](#)
 two-way communication, [1-12](#)

normal status, [3-15](#)

O

objects
 broker configuration, [3-1](#)
 disabling, [8-16](#)

observer, [6-17](#)
 and FastStartFailoverThreshold property, [6-26](#)
 connecting to the configuration, [8-12](#)
 connection to the local computer, [6-52](#)
 detecting
 database shutdown, [6-34](#), [6-38](#)
 instance failures, [6-34](#)

observer (*continued*)
 failures, [6-54](#)
 finding host machine name, [6-52](#)
 installing Oracle Instant Client kit, [2-1](#), [10-97](#)
 lost connection, [6-54](#)
 maintaining fast-start failover configuration information, [6-57](#)
 monitoring, [6-34](#)
 using Oracle Wallet for connection credentials, [10-17](#)

ObserverConnectIdentifier property, [11-37](#)

ObserverOverride property, [11-10](#)

ObserverPingInterval property, [11-11](#)

ObserverPingRetry property, [11-11](#)

ObserverReconnect property, [11-12](#)

operations
 complete failover, [10-49](#)
 disable broker management
 effect on protection modes, [4-30](#)
 downgrade
 effect on protection modes, [4-28](#)
 effect on protection modes, [4-28](#)
 enable broker management
 effect on protection modes, [4-30](#)
 failover, [8-26](#)
 immediate failover, [10-49](#)
 manual failover
 effect on protection modes, [4-29](#)
 removing a database from the configuration
 effect on protection modes, [4-30](#)

OperationTimeout property, [11-12](#)

ORA-16661 message
 reinstating a database, [6-15](#)

ORA-16795 message
 re-creating a database, [6-15](#)

Oracle Clusterware, [1-6](#)
 and instances of an Oracle RAC database, [1-6](#)
 integration with Data Guard broker, [1-6](#)
 recover failed instances, [1-6](#)

Oracle Enterprise Manager, [1-10](#)
 Add Standby Database wizard, [1-10](#)
 database property pages, [1-11](#)
 event management system, [1-6](#)
 integration, [1-10](#)
 integration with the Data Guard monitor, [1-10](#)
 introduction, [1-3](#), [1-9](#)
 making Oracle Net Services configuration changes, [1-6](#), [1-11](#)
 monitoring events, [1-4](#)
 performance tools and graphs, [1-11](#)
 wizards
 creating standby databases, [1-10](#)

Oracle Instant Client
 installing, [2-1](#), [10-97](#)

Oracle Net Services
 configuration changes, [1-6](#), [1-11](#)
 supported configuration, [3-1](#)
 two-way communication, [1-12](#)

Oracle RAC One Node
 supported by Oracle Data Guard broker, [1-2](#)

Oracle Real Application Clusters
 and log apply services, [4-21](#)
 and redo transport services, [4-16](#)
 and setting the apply instance, [4-21](#)
 availability of instances with Oracle Clusterware, [1-6](#)
 benefits, [1-6](#)
 fast-start failover, [6-35](#)
 observer behavior in, [6-35](#)
 observer connection to instances, [6-52](#)
 Oracle Clusterware, [1-6](#)

Oracle Universal Installer
 installing Oracle Instant Client, [2-1](#), [10-97](#)

Oracle Wallet
 starting the observer, [10-17](#)

P

parallel apply
 and log apply services, [4-20](#)
 managing in a physical standby database, [4-20](#)

performance
 Oracle Enterprise Manager tools, [1-11](#)

physical standby databases
 managing parallel apply, [4-20](#)
 state transitions, [4-3](#)
 switchover to the primary role, [6-4](#)

PreferredApplyInstance property, [11-37](#)

PREPARE DATABASE FOR DATA GUARD
 command, [10-60](#)

prerequisites
 installation, [2-2](#)
 switchover, [6-5](#)

primary database
 ARCHIVELOG mode, [2-2](#)
 constructing a standby database, [3-11](#), [8-1](#)
 Data Guard configuration, [3-1](#)
 during failover, [1-11](#)
 during switchover, [6-7](#)
 Flashback Database, [6-15](#)
 health check, [4-37](#)
 preparing for switchover, [6-5](#)
 reinstating after a fast-start failover, [6-62](#)
 state transitions, [4-3](#)
 switching over to the standby role, [10-107](#)

PrimaryLostWriteAction property, [11-14](#)

processes
 DMON, [1-12](#)

processes (*continued*)
 Oracle database, [1-12](#)

properties, [4-7](#)
 configurable, [4-5](#)
 database, [4-5](#)
 managing, [1-14](#)
 monitorable, [4-5](#)
 setting, [3-11](#), [8-5](#)
 in server parameter file, [1-14](#), [2-2](#)
 showing values for, [10-76](#)

property pages
 database, [1-11](#)

protection modes, [6-11](#)
 after a failover, [4-29](#), [6-11](#)
 benefits, [1-6](#)
 configuration, [10-27–10-29](#)
 downgrading, [4-28](#)
 redo transport services setup, [11-34](#)
 setting for a broker configuration, [4-24](#)
 updating, [3-11](#)
 upgrading, [4-28](#)

Q

QUIT command, [10-48](#), [10-63](#)
 See also EXIT command

R

RecvQEntries property, [11-19](#)

Redo Apply, [3-1](#)

redo transport services
 ASYNC mode, [4-9](#)
 configuring, [1-11](#)
 data protection modes, [11-34](#)
 Data Guard configuration, [3-1](#)
 FASTSYNC mode, [4-9](#), [11-34](#)
 in an Oracle RAC environment, [4-16](#)
 managing, [4-8](#)
 SYNC mode, [4-9](#)
 tuning, [4-15](#)
 turning off, [4-13](#)
 turning on, [4-13](#)
 verifying, [1-6](#)

RedoCompression property, [11-39](#)

RedoRoutes property, [11-39](#)

REINSTATE DATABASE command, [10-63](#)

reinstatement, [6-62](#)
 failure logged in broker drc* log files, [6-42](#)
 Flashback Database requirement, [6-15](#)
 flashback logs requirement, [6-15](#)
 how to, [6-15](#)
 logged in broker log files, [6-63](#)
 ORA-16661 message, [6-15](#)
 using DGMGRL, [6-15](#)

reinstatement (*continued*)
 using Enterprise Manager, [6-15](#)

REMOVE CONFIGURATION command, [10-64](#)
 example, [8-20](#)

REMOVE DATABASE command, [10-65](#)

REMOVE FAR_SYNC command, [10-66](#)

REMOVE INSTANCE command, [10-67](#)

REMOVE PLUGGABLE DATABASE command,
[10-67](#)

removing, [10-64](#)
 a standby database, [4-30](#)
 See also each REMOVE command

ReopenSecs property, [11-43](#)

requests
 passing between sites, [1-12](#)

restarting
 databases
 when fast-start failover is enabled, [6-63](#)

role transitions
 changing, [3-11](#)
 managing, [6-1](#)
 fast-start failover, [6-17](#)
 manual failover, [6-8](#)
 switchover, [6-4](#)

S

scripts
 using Data Guard command-line interface
 (DGMGRL), [10-1](#)

selecting the apply instance, [4-21](#)

SendQEntries property, [11-20](#)

server parameter file
 broker property management, [1-14](#), [2-2](#)
 filenames, [1-14](#)
 inconsistent values from Data Guard
 configuration file, [11-17](#)

server parameter files
 validating before role changes, [4-39](#)

server-side software, [1-11](#)

SET ECHO command, [10-69](#)

SET FAST_START FAILOVER TARGET
 command, [10-70](#)

SET STATE command
 and setting the apply instance, [4-21](#)

SET TRACE_LEVEL command, [10-75](#)

setting
 configuration protection mode, [4-24](#)
 database properties, [8-5](#)
 fast start failover target, [10-70](#)
 log apply services, [4-14](#)
 redo transport services, [4-15](#)

SHOW ALL command, [10-76](#)

SHOW CONFIGURATION command, [8-8](#), [10-76](#),
[10-79](#)

SHOW DATABASE command, [10-81](#)

SHOW FAR_SYNC command, [10-85](#)

SHOW FAST_START FAILOVER command,
[10-86](#)

SHOW INSTANCE command, [10-88](#)

SHOW PLUGGABLE DATABASE, [10-92](#)

showing
 See each SHOW command

shutdown abort, [6-34](#)
 effect on fast-start failover, [6-38](#)

SHUTDOWN command, [10-95](#)

shutting down an Oracle instance, [10-95](#)

single command mode
 for Data Guard command-line interface, [10-1](#)

SQL Apply, [3-1](#)

SQL command, [10-97](#)

SQL statements
 executing from DGMGRL command line,
[10-97](#)

standby databases
 choosing a target standby database, [6-2](#)
 constructing from backups, [3-11](#), [8-1](#)
 creating, [1-10](#)
 health check, [4-37](#)
 not involved in a switchover, [6-7](#)
 reenabling after failover, [6-12](#)
 removing, [4-30](#)
 specifying the location of archived redo logs,
[4-14](#)
 switching over to the primary role, [10-107](#)

standby redo log files
 setting up with DGMGRL, [4-26](#)

StandbyAlternateLocation property, [11-43](#)
 setting log apply services, [4-14](#)

StandbyArchiveLocation property, [11-44](#)
 setting log apply services, [4-14](#)

START OBSERVER command
 fast-start failover configuration file (fsfo.dat),
[6-57](#)

starting, [10-1](#)
 Data Guard command-line interface
 (DGMGRL), [8-2](#)
 Data Guard monitor (DMON), [3-10](#)
 manual failover, [6-10](#)
 Oracle instance, [10-103](#)
 switchover, [6-6](#)

STARTUP command, [10-103](#)

state transitions
 effect on database states, [4-3](#)
 logical standby database, [4-3](#)
 physical standby database, [4-3](#)
 primary database, [4-3](#)

states, [3-14](#)
 changing, [8-15](#)
 database, [4-1](#)

states (*continued*)
 database transitions, [4-3](#)

static connect identifiers
 validating, [10-127](#)
 validating before role changes, [4-40](#)

StaticConnectIdentifier property, [11-45](#)

status
 configuration, [3-15](#)
 health check on primary database, [4-37](#)
 health check on standby database, [4-37](#)
 health of the database, [4-37](#)
 intended state of a configuration, [3-15](#)

status messages
 ORA-16661, [6-15](#)
 ORA-16795, [6-15](#)

STOP OBSERVER command, [10-97](#), [10-105](#)

string values
 Data Guard command-line interface, [10-7](#)

SWITCHOVER command, [8-21](#), [10-107](#)

SWITCHOVER PLUGGABLE DATABASE
 command, [10-112](#)

switchovers
 benefits, [1-6](#)
 broker tasks, [6-7](#)
 choosing a target standby database, [6-2](#)
 DGMGRL SWITCHOVER command, [8-21](#)
 effect on
 database startup, [10-107](#)
 primary database, [6-7](#)
 protection modes, [4-29](#)
 standby databases not involved in the
 switchover, [6-7](#)

managing, [6-1](#), [6-4](#)
 overview, [6-1](#)
 Oracle Enterprise Manager, [1-11](#)
 performing when fast-start failover is
 enabled, [6-38](#)
 preparing the primary database, [6-5](#)
 prerequisites, [6-5](#)
 reenabling disabled standby databases, [6-15](#)
 starting, [6-6](#)
 transitioning
 a logical standby database to the primary
 role, [6-4](#)
 a physical standby database to the
 primary role, [6-4](#)
 using the DGMGRL SWITCHOVER
 command, [10-107](#)

SYNC redo transport mode, [4-9](#)

SYSDBA privileges
 to connect to the database, [10-7](#)

T

target standby database
 choosing, [6-2](#)
 disabling fast-start failover on, [6-45](#)

TopWaitEvents property, [11-22](#)

TraceLevel property, [11-15](#)

transport lag, [4-16](#)

TRANSPORT-ON state
 setting LogShipping property, [11-33](#)

TransportDisconnectedThreshold property, [11-46](#)

TransportLagThreshold property, [11-46](#)

troubleshooting
 Data Guard, [12-1](#)
 diagnostics logged in broker drc* log files,
[12-1](#)

tuning
 redo transport services, [4-15](#)

tuning fast-start failover, [4-33](#)

two-way communication channel, [1-12](#)

U

unobserved configuration, [6-54](#)

updating configuration properties, [4-5](#)

upgrading
 protection mode, [4-28](#)

user interfaces
 overview, [1-10](#)

V

V\$DATABASE view
 fast-start failover columns, [6-36](#)
 FS_FAILOVER_OBSERVER_HOST column,
[6-52](#)
 FS_FAILOVER_OBSERVER_PRESENT
 column, [6-52](#)
 viewing
 fast-start failover statistics, [6-41](#)

V\$DATAGUARD_PROCESS
 using to monitor broker processes, [1-12](#)

V\$FS_FAILOVER_STATS view, [6-44](#)

VALIDATE DATABASE command, [10-113](#)

VALIDATE DATABASE SPFILE command, [4-39](#),
[10-118](#)

VALIDATE DGConnectIdentifier command,
[10-120](#)

VALIDATE FAR_SYNC command, [10-122](#)

VALIDATE NETWORK CONFIGURATION
 command, [4-40](#), [10-125](#)

VALIDATE STATIC CONNECT IDENTIFIER
 command, [4-40](#), [10-127](#)

W

warning status, [3-15](#)

wizards

Add Standby Database, [1-10](#)