

Data Guard Installation and Configuration

Oracle Database 10g

Table of Content

1. INTRODUCTION	3
1.1. PURPOSE OF DOCUMENT	3
1.2. INTENDED AUDIENCE	3
1.3. SCOPE OF THIS GUIDE	3
2. DATA GUARD INSTALLATION	4
2.1. PREPARATION	4
2.1.1. ENABLE FORCE LOGGING	4
2.1.2. CREATE A PASSWORD FILE	4
2.1.3. CONFIGURE STANDBY REDO LOG	5
2.1.4. SET PRIMARY DATABASE INITIALIZATION FILE	6
2.1.5. SET ORACLE NET SERVICE FILES	8
2.1.6. ENABLE ARCHIVING	8
2.2. CREATING PHYSICAL STANDBY DATABASE	9
2.2.1. CREATE BACKUP COPY PRIMARY DATABASE	9
2.2.2. CREATE CONTROL FILE FOR STANDBY DATABASE	9
2.2.3. PREPARE AND COPY INITIALIZATION FILE TO STANDBY DATABASE	9
2.3. STARTING PHYSICAL STANDBY DATABASE	10
2.4. TESTING PHYSICAL STANDBY DATABASE	10
3. MANAGING DATA GUARD	11
3.1. SHUTTING DOWN DATA GUARD AND DATABASE INSTANCE	11
3.2. STARTING UP DATA GUARD AND DATABASE INSTANCE	11
3.3. MANUAL REDO LOG GAP RESOLUTIONS	12
3.4. HANDLING REDO LOG SHIPPING THAT STUCKED	12
3.5. SIGNIFICANT REDO LOG GAP RESOLUTIONS	14
4. SWITCH OVER SCENARIO	16
5. FAIL OVER SCENARIO	19
6. ACTIVATE STANDBY FOR TESTING SCENARIO	22

1. INTRODUCTION

1.1. Purpose of Document

This document provides technical information about installing and configuring Oracle Database Data Guard.

1.2. Intended Audience

This manual is intended for

- IT / Support department
- Technical staffs that perform maintenance on Oracle Database.

1.3. Scope of this guide

This document consists of 2 parts:

- The first part is to list down the preparation to install Oracle Data Guard.
- The second part is to describe the steps to install and configure Oracle Data Guard.
- The third part is to describe the steps to do switch over and fail over mechanism in Oracle Data Guard.

2. Data Guard Installation

2.1. Preparation

- Use Oracle Database version 10 or more.
- Makes sure the mounting point and oracle path are the same between Primary and Standby database
- There are several things that need to be done in the Primary server, before making a cold backup to the standby server.

2.1.1. Enable Force Logging

Login as oracle user to the Oracle database server, then type these commands:

```
bash
sqlplus sys as sysdba
SQL> ALTER DATABASE FORCE LOGGING;
```

This command might take some times to complete, because it will wait unlogged direct write I/O to finish.

2.1.2. Create a Password File

Create a password file if one does not already exist. Every database in a Data Guard configuration must use a password file, and the password for the `SYS` user must be identical on every system for redo data transmission to succeed.

If password file does not exist then it can be created by logging in as oracle user and use these commands:

```
bash
orapwd file=filename password=password entries=max_users
```

filename for unix usually `$ORACLE_HOME/dbs/orapwSIDNAME.ora`

filename for windows usually `$ORACLE_HOME/database/PWDSIDNAME.ora`

password fill that entry with the desired password

max_users fill that entry with number that will represent the maximum number of user that will be granted as SYSDBA

Besides using that command also add one entry in the init files of the database, it can be done by typing these commands:

```
bash
cd $ORACLE_HOME/dbs
vi initSIDNAME.ora
```

append this line of code in the end of the file

```
*.remote_login_passwordfile='EXCLUSIVE'
```

2.1.3. Configure Standby Redo Log

A standby redo log is required for the maximum protection and maximum availability modes. Standby Redo Log must exist since it will be used when the primary become the standby database, since standby database will use Standby Redo Log rather than Online Redo Log.

In creating Standby Redo Log (SRL), consider this:

- The size of the Standby Redo Log must be the same with Online Redo Log
- The number of the Standby Redo Log must us this formula = sum of all production online log groups for each thread + number of threads. For example, if a primary database has two instances (threads) (RAC) and each thread has four online log groups, then there should be ten SRLs.
- Must exist in both Primary and Standby databases
- Should be created in a protected disk, redundancy.
- In RAC, must be placed on a shared disk
- In RAC, assign the SRL to a thread / instance when the SRL is created.
- Consider the value of MAXLOGFILES and MAXLOGMEMBERS in CREATE DATABASE statement. This value will limit the number of Online Redo Log and Standby Redo Log Groups and Files. To check use this command in SQL PLUS

```
SQL> alter database backup controlfile to trace as 'datapath';
```

To create Standby Redo Log it can be done by doing this:

- In RAC environment, SRL should be assigned to specific thread, thread is representing the instance in Primary RAC, if not specified this will be assigned automatically.

```
SQL> alter database add standby logfile thread threadNumber
2> ('/oracle/dbs/log1c.rdo', '/oracle/dbs/log2c.rdo') size 500M;
```

- In Non RAC environment, SRL should be assigned into Group of Redo Log. The numbering plan must follow the current Group number. However usually from number 1 to 4 already being used as Online Redo Log. The script is

```
SQL> alter database add standby logfile group groupNumber
2> ('/oracle/dbs/log1c.rdo', '/oracle/dbs/log2c.rdo') size 500M;
```

If somehow in standby database there will be changes regarding Standby Redo Log or Online Redo Log, make sure recovery process is not running and the database is in mounted position, also change parameter **STANDBY_FILE_MANAGEMENT** to **MANUAL**, after the changes is done change it back to **AUTO**

Verify the Standby Redo Log file groups were created, by using this command

```
SQL> select group#, thread#, sequence# archived, status from v$standby_log;
```

2.1.4. Set Primary Database Initialization File

On the primary database, initialization parameters being defined to controls redo transport services while the database is in the primary role. However, there are additional parameters that are need to add that control the receipt of the redo data and log apply services when the primary database is transitioned to the standby role.

As an example, a scenario is created. Consider, one database instance that is called as jakarta, that functioned as primary database. While another database instanced named as bandung, functioned as standby database. Consider both instance are hosted in different servers. The standby server is going to be made as Physical Standby Server.

Database	DB_UNIQUE_NAME	Oracle Net Service Name
Primary	jakarta	jakarta
Physical Standby	bandung	bandung

Part of the initialization file that relates to Data Guard configuration for Primary Database might look like this:

```
DB_NAME=jakarta
DB_UNIQUE_NAME=jakarta
CONTROL_FILES='/arch1/control1.ctl','/arch2/control2.ctl'
FAL_CLIENT=jakarta
FAL_SERVER=bandung
LOG_ARCHIVE_FORMAT='%t_%s_%r.dbf'
LOG_ARCHIVE_CONFIG='DG_CONFIG=(jakarta,bandung)'
LOG_ARCHIVE_DEST_1=
  'LOCATION=/arch1
  VALID FOR=(ALL LOGFILES,ALL_ROLES)
  DB_UNIQUE_NAME=jakarta'
LOG_ARCHIVE_DEST_2=
  'SERVICE=bandung
  VALID FOR=(ALL LOGFILES,PRIMARY_ROLE)
  DB_UNIQUE_NAME=bandung LGWR ASYNC REOPEN=30'
REMOTE_LOGIN_PASSWORDFILE=EXCLUSIVE
STANDBY_FILE_MANAGEMENT=AUTO
LOG_FILE_NAME_CONVERT='/oraredo1','/oraredo1','/oraredo2','/oraredo2'
```

Part of the initialization file that relates to Data Guard configuration for Physical Standby Database might look like this:

```
DB_NAME=jakarta
DB_UNIQUE_NAME=BANDUNG
CONTROL_FILES='/arch1/standby1.ctl','/arch2/standby2.ctl'
FAL_CLIENT=bandung
FAL_SERVER=jakarta
LOG_ARCHIVE_FORMAT='%t_%s_%r.dbf'
LOG_ARCHIVE_CONFIG='DG_CONFIG=(jakarta,bandung)'
LOG_ARCHIVE_DEST_1=
  'LOCATION=/arch1
  VALID FOR=(ALL LOGFILES,ALL_ROLES)
  DB_UNIQUE_NAME=bandung'
LOG_ARCHIVE_DEST_2=
  'SERVICE=jakarta
  VALID FOR=(ALL LOGFILES,PRIMARY_ROLE)
  DB_UNIQUE_NAME=jakarta LGWR ASYNC REOPEN=30'
REMOTE_LOGIN_PASSWORDFILE=EXCLUSIVE
STANDBY_FILE_MANAGEMENT=AUTO
LOG_FILE_NAME_CONVERT='/oraredo1','/oraredo1','/oraredo2','/oraredo2'
```

Below is the explanation that is used in Data Guard configuration:

- DB_NAME, specify the primary database instance name, apply the same name for all standby database.
- DB_UNIQUE_NAME, specify unique name for each database, for primary the value will be the same as DB_NAME, while for standby database the value must be different. It is preferable if DB_UNIQUE_NAME is as the same as Oracle Net Service Name.
- LOG_ARCHIVE_CONFIG, will define the list of database in Data Guard configuration, so that log transport will be allowed between the database that is listed. The value that is used DB_UNIQUE_NAME. This parameter will enable the switchover without having to change parameter to defer / enable destinations. For RAC implementation, after role transition this setting need to be set again using SEND, NOSEND, RECEIVE, or NORECEIVE.
- CONTROL_FILE, specify the path to control files in primary database, while in standby database will specify the path to standby control files. It is recommended to duplicate control files and standby control files.
- FAL_SERVER, Specify the Oracle Net service name of the FAL server (typically this is the database running in the primary role). When the bandung database is running in the standby role, it uses the jakarta database as the FAL server from which to fetch (request) missing archived redo log files if jakarta is unable to automatically send the missing log files.
- FAL_CLIENT, Specify the Oracle Net service name of the Boston database. The FAL server (jakarta) copies missing archived redo log files to the bandung standby database.
- LOG_ARCHIVE_FORMAT, specify the format for the archived redo log files using a thread (%t), sequence number (%s), and resetlogs ID (%r).
- LOG_ARCHIVE_DEST_n, specify where the redo data is to be archived on the primary and standby systems.
 - LOG_ARCHIVE_DEST_1 archives redo data generated by the primary database from the local online redo log files to the local archived redo log files in /arch1.
 - LOG_ARCHIVE_DEST_2 is valid only for the primary role. This destination transmits redo data to the remote physical standby destination bandung. There are several parameters that are used
 - LGWR ASYNC, everytime write event in primary into online redo log, asynchronously transmitted to the standby database into standby redo log
 - REOPEN, spcify the minimum number of seconds before redo transport services should try to reopen a failed destination.
- REMOTE_LOGIN_PASSWORDFILE, set the same password for SYS on both primary and standby databases. The recommended is either EXCLUSIVE or SHARED. By specifying this parameter, a secure redo data transmission is provided.
- STANDBY_FILE_MANAGEMENT, Set to AUTO so when datafiles are added to or dropped from the primary database, corresponding changes are made automatically to the standby database.
- STANDBY_ARCHIVE_DEST, set this if in the standby database, the directory to store archive log is different than the primary database.
- LOG_FILE_NAME_CONVERT, specify the location of primary database online redo log files followed by the location of online redo log in standby database. This parameter usefull when the location of online redo log different between primary and standby database. However this parameter should be use when the directory are the same. Multiple pairs of paths may be specified by this parameter.

Review the initialization parameter file for additional parameter that may need to be modified, for example AUDIT_FILE_DEST.

2.1.5. Set Oracle Net Service Files

Oracle Net Service consisted of two files; tnsnames.ora and listener.ora. For listener.ora make it look like this

```
SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (SID_NAME = jakarta)
      (ORACLE_HOME = /u01/oracle)
      (GLOBAL_NAME = jakarta)
    )
  )

LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = TCP) (HOST = IPPrimary) (PORT = 1521))
    )
  )

ADR_BASE_LISTENER = /u01/oracle
```

While for tnsnames.ora it will look like:

```
bandung =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP) (HOST = IPStandby) (PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = jakarta)
    )
  )

jakarta =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP) (HOST = IPPrimary) (PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = jakarta)
    )
  )
```

2.1.6. Enable Archiving

If archiving is not enabled, issue the following statements to put the primary database in ARCHIVELOG mode and enable automatic archiving:

```
SQL> SHUTDOWN IMMEDIATE;
SQL> STARTUP MOUNT;
SQL> ALTER DATABASE ARCHIVELOG;
SQL> ALTER DATABASE OPEN;
```

2.2. Creating Physical Standby Database

2.2.1. Create Backup Copy Primary Database

To create a backup copy of Primary database, the mechanism that will be used is Cold Backup. A cold backup require the database to be shutdown (not a shutdown abort). Then copy all the necessary files to the destined place. Below is the list of files that must be copied:

- All data files
- All redo logs
- Password file that located in \$ORACLE_HOME/dbs
- tnsnames.ora and listener.ora that located in \$ORACLE_HOME/network/admin

Another mechanism is to use RMAN backup from the Primary database, this method will make no downtime in primary database when creating standby database. The steps are:

- Create full backup of datafiles and archive log (See Oracle doc Backup and Recovery Advanced Guide), also create standby controlfile (See step 2.2.2)
- Copy those backup files to the machine that will host standby database, using the same directory structure as in Primary database
- Edit init files for standby database by comment out control_files parameter
- Start standby database using command "startup nomount pfile=/location/pfile;"
- From Primary database connect to the RMAN using this command "rman target sys/password@primary nocatalog auxiliary sys/password@standby"
- Inside RMAN use this command to start the process "duplicate target database for standby nofilenamecheck;"
- After the process complete standby database will be in mounted position, check and make necessary changes on the location of Online Redo Log, Standby Redo Log, and Standby control file.
- After all the checking is done applying redo log can be started.

2.2.2. Create Control File for Standby Database

The creation of Control File can be continued after creating backup for Primary Database, the next step that can be done are:

```
SQL> alter database create standby controlfile as '/tmp/standby1.ctl';
```

The newly created standby controlfile must be copied to the standby server and in init file for physical standby database use this standby controlfile as the controlfile. It is suggested to create more than one standby controlfile by copying from the first one.

2.2.3. Prepare and Copy Initialization File to Standby Database

To create initialization file from the Database server, type this command:

```
SQL> create pfile='/tmp/initSIDNAME.ora' from spfile;
```

Edit the value of the init file so that the parameters part that are used in Data Guard configuration are the same as the above part. Make sure while editing the init file, every directories or files that are stated there, are existed. After that copy the edited init file to the standby server in \$ORACLE_HOME/dbs directory.

After the initialization file being copied to the standby database server then server parameter file can be created for physical standby database server. Just login to standby database server using oracle user, then type this command in sqlplus' idle session

```
SQL> create spfile from pfile='/the/location/of/init/file/initSIDNAME.ora;
```

Now the Physical Standby Server is ready.

2.3. Starting Physical Standby Database

To start Physical Standby database, login to the standby server, and type these commands:

```
sqlplus sys as sysdba
SQL> startup nomount;
SQL> alter database mount standby database;
```

After starting Physical Standby database server, issue this command in SQL Plus to start Redo Apply process.

```
SQL>alter database recover managed standby database disconnect from session;
```

DISCONNECT FROM SESSION options means that Redo Apply process will run in background process.

2.4. Testing Physical Standby Database

To test if physical standby server is functioned or not, it can be done by starting the transmission of redo data to the remote standby location. This does not occur until after a log switch. A log switch occurs, by default, when an online redo log file becomes full. To force a log switch so that redo data is transmitted immediately, use the following ALTER SYSTEM statement on the primary database. For example:

```
SQL> ALTER SYSTEM SWITCH LOGFILE;
```

To check whether the Redo Transport is succeeded or not use this command in primary,

```
SQL> SELECT SEQUENCE#, FIRST_TIME, NEXT_TIME
       FROM V$ARCHIVED_LOG ORDER BY SEQUENCE#;
```

To check whether the Redo Apply is succeeded or not use this command in standby,

```
SQL> SELECT SEQUENCE#,APPLIED FROM V$ARCHIVED_LOG
       ORDER BY SEQUENCE#;
```

Make sure if both last sequence number are the same and APPLIED is YES. If NO use this query to check last applied log:

```
SQL> SELECT THREAD#, MAX(SEQUENCE#) AS "LAST_APPLIED_LOG"
       FROM V$LOG_HISTORY
       GROUP BY THREAD#;
```

To Check Standby DB status :

```
SQL>SELECT DATABASE_ROLE, DB_UNIQUE_NAME INSTANCE,
       OPEN_MODE, PROTECTION_MODE, PROTECTION_LEVEL,
       SWITCHOVER_STATUS FROM V$DATABASE;
```

3. Managing Data Guard

Sometimes Data Guard need to be restarted, this might happen if the database wants to be upgraded, or many things else.

3.1. Shutting Down Data Guard and Database Instance

In the Primary Database server type these codes

```
SQL> alter system switch logfile;
```

```
SQL> alter system archive log current;
```

```
SQL> alter system set log_archive_dest_state_2=defer scope=both;
```

```
SQL> shutdown immediate;
```

In the Physical Standby Database server type this code

```
SQL> alter database recover managed standby database cancel;
```

```
SQL> shutdown immediate;
```

After this either Primary or Physical Standby database server can shutted down.

3.2. Starting Up Data Guard and Database Instance

In the Primary Database server type this command

```
SQL> startup open;
```

```
SQL> alter system set log_archive_dest_state_2=enable scope=both;
```

In the Physical Standby Database server type this code, consider that the instance already being mounted as Standby database

```
SQL> startup nomount;
```

```
SQL> alter database mount standby database;
```

```
SQL> alter database recover managed standby database using current logfile  
disconnect from session;
```

OR

```
SQL> alter database recover managed standby database disconnect from  
session;
```

To test wether the Data Guard is already configured or not, do the steps in chapter 2.4.

3.3. Manual Redo Log Gap Resolutions

Redo Log Gap usually done automatically based on one of the parameter in the initialization, `FAL_SERVER` and `FAL_CLIENT`. However, in some cases, automatic gap recovery may not take place and it has to be done manually.

Perform the following query at the physical standby database to determine if there is redo gap on a physical standby database:

```
SQL> select * from v$archive_gap;
```

The output from the previous codes will show that the physical standby database is currently missing log files (if exist) from sequence `LOW_SEQUENCE#` to sequence `HIGH_SEQUENCE#` for thread `THREAD#` (those are the columns of `v$archive_gap`).

In Primary database do this query to check the location of the missing logs in Primary server:

```
SQL> select NAME from v$archive_log where THREAD#=THREAD# and  
DEST_ID=1 and SEQUENCE# between LOW_SEQUENCE# and  
HIGH_SEQUENCE#;
```

Replace the value of **THREAD#**, **LOW_SEQUENCE#**, **HIGH_SEQUENCE#** from the previous query that was resulted in Physical Standby database. While `DEST_ID=1` means that local archive log are stored in `log_archive_dest_1`.

After the missing archive logs are found, copy those archive logs to the standby server, and place them in the place where archive log suppose to be stored. Then register those missing archive log using this code in the Physical Standby database:

```
SQL> alter database register physical logfile '/location/missing/archive/log';
```

Register each of the missing logs to the Physical Standby database using this code.

After finish applying all the missing archive logs, repeat the steps from the beginning to check whether is there any other Redo Log Gap.

3.4. Handling Redo Log Shipping that Stucked

Redo log shipping process can be stucked whenever there are network disruption between Primary database and Standby database. Several thing that need to be done in order to start the redo log shipping again :

Find arc process in the primary database, note that arc process from oracle are numbered according to `log_archive_dest_n`, so `arc0` will belongs to `log_archive_dest_1` and `arc1` belongs to `log_archive_dest_2`, and so on. Usually `log_archive_dest_state_2` is configured as destination to the standby database, and when the destination is having network disruption the `arc1` process will hang. So `arc1` process must be killed using signal 9, killing archiver process will make oracle engine start a new archiver process by itself.

```
ps -ef | grep arc1  
kill -9 pid-of-arc1
```

In the alert log there will be record like this, after killing arc1 process.

```
ARCH: Detected ARCH process failure
ARCH: STARTING ARCH PROCESSES
ARC1: Archival started
ARCH: STARTING ARCH PROCESSES COMPLETE
ARC1 started with pid=124, OS id=60
ARC1: Becoming the heartbeat ARCH
```


Check the arc1 process using `ps -ef`, it should have new PID

```
ps -ef | grep arc1
```

After the archiver process in the primary database killed, there will be archive log gap in the standby database, and have to be resolved. It can be checked using this query

```
SQL> SELECT SEQUENCE#,APPLIED FROM V$ARCHIVED_LOG
ORDER BY SEQUENCE#;
```

It will show result such as :



SEQUENCE#	APP
23438	NO
23439	NO
23440	NO
23441	NO
23442	NO
23443	NO
23444	NO
23445	NO
23446	NO
23447	NO
23448	NO
23450	NO
23451	NO
23453	NO
23454	NO
23455	NO
23456	NO
23457	NO

Note that the result of the query show if there is missing sequence 23449, this means that archive log with that sequence is not in the standby database. While applied column shows a lot of NO, this is because there are missing sequence in the standby database.

By killing the arc1 process in the primary database, primary database will start synching the archive log again (consider that the network diruption is fixed). If there are missing archive log that haven't send to the standby database, method in 3.3 Manual Redo Log Gap Resolution need to be done.

To know which archive log that have been sent to the standby database, tail alert log file

This will show that archive log sequence 23341 is received at the standby database

```
RFS[796]: Archived Log:
'/xanadu/oradata/arc/1_23341_685042871.dbf'
```

While this log will show that archive log with sequence 23330 is applied

```
Media Recovery Log /xanadu/oradata/arc/1_23330_685042871.dbf
```

Applying redo log always take longer time than receiving archive log, so that's why the main task that needs to be done is to make sure the archive log gap is resolved first.

3.5. Significant Redo Log Gap Resolutions

This method can be used whenever there are big gaps in redo logs between primary and standby database (one month difference in active OLTP database is huge). This means that the method in 3.3 Manual Redo Log Gap Resolutions can be done over hundred of times, and this is unacceptable. The solution to this matter is to apply the incremental backup of primary from that SCN number.

On the standby database check the current scn

```
SQL> select current_scn from v$database;
```

The result can be compared by the same query in the primary database, to convert scn into date, it can be done using this query

```
SQL> select scn_to_timestamp(obtained-scn) from dual;
```

After the current scn from standby is obtained, then an incremental backup from primary needed . Login into RMAN console in the primary database

```
RMAN target /
RMAN> run {
2> allocate channel c1 type disk format '/path/arc/log';
3> backup incremental from scn obtained-scn database;
4> }
```

After that a standby control file is needed from primary database

```
SQL> alter database create standby controlfile as
'/path/standby/ctrl/file';
```

After Incremental backup and standby control file are created, copy those files to the standby database, in Unix or Linux can use scp command

```
scp -r /dir/to/be/copied username@ip-dest:/dir/destination
```

In the standby database shutdown the redo log apply and the instance too

```
SQL> alter database recover managed standby database cancel;
SQL> shutdown immediate;
```

Startup nomount the standby database then check the location of control files

```
SQL> startup nomount;
SQL> show parameter control_files;
```

From the result of the above query, replace the control files with the standby control file that have just created and copied from the primary database. It would be better if you save a copy from the older standby control files. If the replacement of the standby control files is finished, the standby database can be mounted.

```
SQL> alter database mount standby database;
```

After the standby database is mounted, RMAN need to know the location of the copied incremental backup from primary database

```
RMAN target /
RMAN> catalog start with '/path/incrmnt1/backup';
```

Just type YES when RMAN asking about catalogin files. When the catalog process is finished the recovery process can be started in RMAN

```
RMAN> recover database;
```

After some time, the recovery fails with the message :

```
RMAN-00571:
=====
RMAN-00569: ===== ERROR MESSAGE STACK FOLLOWS
=====
RMAN-00571:
=====
RMAN-03002: failure of recover command at 12/18/2009 06:53:02
RMAN-11003: failure during parse/execution of SQL statement: alter
database recover logfile '/u01/oradata/1_8008_697108460.dbf'
ORA-00310: archived log contains sequence 8008; sequence 8009
required
ORA-00334: archived log: '/u01/oradata/1_8008_697108460.dbf'
```

This happens because we have come to the last of the archived logs. The expected archived log with sequence# 8008 has not been generated yet. It can be ignored, and redo log apply process can be started again in the standby database.

```
SQL> alter database recover managed standby database using
current logfile disconnect from session;
```

To confirm if the process is success or not it can be done by querying current_scn from standby and primary database, the result shouldn't be very different. The rest of the process is to monitor whether the redos are shipped and applied or not, please refer to 2.4 Testing Physical Standby Database.

4. Switch Over Scenario

Switchover scenario will consider two server that host the database server is in healthy condition and the database instance is healthy too.

This section describes how to perform a switch over to a physical standby database. A switchover is initiated on the primary database and is completed on the target standby database.

This process requires the Data Guard to be stopped first, see chapter 3.1

Step 1 : Verify the status of the Primary database to see if it can be switched to the standby role, it can be done by this query.

```
SQL> select switchover_status from v$database;  
SQL> alter system switch logfile;  
SQL> alter system archive log current;
```

If the result of the query is TO_STANDBY or SESSIONS_ACTIVE then the Primary Server can be switched to the standby role. If neither of these values is returned, then the switchover process can not be continued. Wait until the sequence of redo log between primary and standby are the same.

The SWITCHOVER_STATUS column of v\$database can have the following values:

NOT ALLOWED - Either this is a standby database and the primary database has not been switched first, or this is a primary database and there are no standby databases.

SESSIONS ACTIVE - Indicates that there are active SQL sessions attached to the primary or standby database that need to be disconnected before the switchover operation is permitted.

SWITCHOVER PENDING - This is a standby database and the primary database switchover request has been received but not processed.

SWITCHOVER LATENT - The switchover was in pending mode, but did not complete and went back to the primary database.

TO PRIMARY - This is a standby database, with no active sessions, that is allowed to switch over to a primary database.

TO STANDBY - This is a primary database, with no active sessions, that is allowed to switch over to a standby database.

RECOVERY NEEDED - This is a standby database that has not received the switchover request.

During normal operations it is acceptable to see the following values for SWITCHOVER_STATUS on the primary to be SESSIONS ACTIVE or TO STANDBY.

During normal operations on the standby it is acceptable to see the values of NOT ALLOWED or SESSIONS ACTIVE.

Step 2 : Initiate the switchover on the primary database, by issuing this SQL statement

```
SQL> alter database commit to switchover to physical standby with session shutdown;
```

This will make the primart database into a physical database, and the current control file is backed up to the current SQL session trace file before switchover, therefore it is possible to reconstruct a current control file, if necessary.

WITH SESSION SHUTDOWN can be omitted if the query performed in the previous step returned a value of TO STANDBY.

Step 3 : Shut down and then mount the former primary database, by typing these codes

```
SQL> shutdown immediate;
SQL> startup nomount;
SQL> alter database mount standby database;
```

At this point in the switchover process, the original primary database is a physical standby database.

Step 4 : Verify that the switchover target is ready to be switched to the primary role, by using this code

```
SQL> select switchover_status from v$database;
```

A value TO_PRIMARY or SESSIONS_ACTIVE indicates that the standby database is ready to be switched to the primary role. If not then it is not ready yet, it might be because Redo Apply is active and redo transport is configured and working. Continue to query until the value returned TO_PRIMARY or SESSIONS_ACTIVE.

Step 5 : Switch the target physical standby database role to the primary role, by issuing the following SQL statement on the targeted physical standby database:

```
SQL> alter database recover managed standby database finish;
SQL> alter database commit to switchover to primary with session shutdown;
```

WITH SESSION SHUTDOWN can be omitted if the result of the previous statement is TO_PRIMARY.

If "ORA-00261: log 12 of thread 1 is being archived or modified" happen use

```
SQL> alter database recover managed standby database finish force;
```

There will be some data loss by issuing this command

Step 6 : Open the new Primary database, by typing this

```
SQL> alter database open;
```

Step 7 : Start Redo Apply on the new Physical Standby Database by using this statement

```
SQL> alter database recover managed standby database using current logfile  
disconnect from session;
```

OR

```
SQL> alter database recover managed standby database disconnect from  
session;
```

Step 8 : Test the new Data Guard configuration by referring chapter 2.4

5. Fail Over Scenario

Failover scenario is used if the primary server is having a problem either the server's Operating System or the database server. There are several steps that need to be done:

Step 1 : Flush any unspent redo from the primary database to the target standby database. This is possible if the primary database can be mounted. If successful it will become a zero data loss failover. To do so ensure that the target standby database is having Redo Apply active.

Mount, but do not open the primary database, if can not skip this to step 2. Issue this SQL statement in the mounted primary database:

```
SQL> alter system flush redo to target_db_name;
```

For **target_db_name** use DB_UNIQUE_NAME of the standby database in its initialization file.

If this statement completes without any errors, go to Step 5. If the statement completes with any error, or if it must be stopped because you cannot wait any longer for the statement to complete, continue with Step 2.

Step 2 : Identify and resolve any gaps in the archived redo log files. To do so query the v\$archive_gap on the target standby database.

```
SQL> select * from v$archive_gap;
```

Then for the next step refer to chapter 3.3 to resolve the gap manually.

Step 3 : Copy any other missing archived redo log files, query the v\$archived_log view on the target standby database to obtain the highest sequence number for each thread, the example is like this

```
SQL> select unique thread# as thread, max(sequence#) over (partition by  
thread#) as last from v$archived_log;
```

Copy any available archived redo log files from the primary database that contains sequence numbers higher than the highest sequence number available on the target standby database to the target standby database and register them. This must be done for each thread. Refer to LOG_ARCHIVE_FORMAT in chapter 2.1.4 to read the format of archive log files. After that register the archive log file to the standby server, using this statement:

```
SQL> alter database register physical logfile '/location/missing/archive/log';
```

Step 4 : Makes sure to repeat step 2 – 3 to make sure no more redo log gap, if the redo log gap can not be resolved, that means there will be some data loss in failover operations.

Step 5 : Initiate failover operations on the target physical standby database by issuing this command:

```
SQL> alter database recover managed standby database cancel;
```

Step 6 : Finish applying all received redo data, by issuing this command:

```
SQL> alter database recover managed standby database finish;
```

If this statement completes without any errors, proceed to Step 7. If an error occurs, some received redo data was not applied. Try to resolve the cause of the error and re-issue the statement before proceeding to the next step.

If "ORA-00261: log 12 of thread 1 is being archived or modified" happen use

```
SQL> alter database recover managed standby database finish force;
```

There will be some data loss by issuing this command

Note that if there is a redo gap that was not resolved in Step 2 and Step 3, there will be an error stating that there is a redo gap. If the error condition cannot be resolved, a failover can still be performed (with some data loss) by issuing the following SQL statement on the target standby database:

```
SQL> alter database activate physical standby database;
```

Then skip to the step 9.

Step 7 : Verify that the target standby database is ready to become a primary database, by querying v\$database view on the target standby database

```
SQL> select switchover_status from v$database;
```

A value of either TO PRIMARY or SESSIONS ACTIVE indicates that the standby database is ready to be switched to the primary role. If neither of these values is returned, verify that Redo Apply is active and continue to query this view until either TO PRIMARY or SESSIONS ACTIVE is returned.

Step 8 : Switch the physical standby database to the primary role by issuing this SQL statement on the target standby database:

```
SQL> alter database commit to switchover to primary with session shutdown;
```

WITH SESSION SHUTDOWN can be omitted if the previous statement resulted TO_PRIMARY value.

Step 9 : Open the new primary database

```
SQL> alter database open;
```

Step 10 : It is recommended to create full backup of the new primary database

Step 11 : Restart Redo Apply if it has stopped at any of the other physical standby databases in the Data Guard configuration (more than one Standby database)

```
SQL> alter database recover managed standby database using current logfile  
disconnect from session;
```

OR

```
SQL> alter database recover managed standby database disconnect from  
      session;
```

Step 12 : Optionally restore the failed primary database, by making it as physical standby database, then perform switchover to restore it to primary role.

6. Testing Scenario

Testing scenario is used whenever the standby database is going to be used as testing database with production data, the mechanism involving flashback capability. The standby database will be activated and after the test is finish, it will be reverted back using flashback

Step 1 : Check that primary and standby are synchronized, by checking the current log sequence, using the archive log list command.

PRIMARY

```
ORACLE_SID=DG
ORACLE_BASE=/u01/oracle
ORACLE_HOME=/u01/oracle/10g
```

```
SQL> archive log list
Database log mode           Archive Mode
Automatic archival         Enabled
Archive destination
USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence 32
Next log sequence to archive 34
Current log sequence       34
```

```
SQL> select database_role, open_mode from
v$database;
```

```
DATABASE_ROLE    OPEN_MODE
-----
PRIMARY          READ WRITE
```

PHYSICAL STANDBY

```
ORACLE_SID=DG
ORACLE_BASE=/u01/oracle
ORACLE_HOME=/u01/oracle/10g
```

```
SQL> archive log list;
Database log mode           Archive Mode
Automatic archival         Enabled
Archive destination
USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence 0
```

```
Next log sequence to archive    0
Current log sequence            34
```

```
SQL> select database_role, open_mode from
v$database;
```

```
DATABASE_ROLE    OPEN_MODE
-----
PHYSICAL STANDBY MOUNTED
```

Step 2 : Check that standby is applying redo logs, on this step we will make switch logfile on the primary and will check how the sequence is advanced also on the physical standby

PRIMARY

```
SQL> alter system switch logfile;
```

System altered.

```
SQL> archive log list;
Database log mode                Archive Mode
Automatic archival                Enabled
Archive destination
USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence       34
Next log sequence to archive     36
Current log sequence              36
```

PHYSICAL STANDBY

```
SQL> archive log list;
Database log mode                Archive Mode
Automatic archival                Enabled
Archive destination
USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence       34
Next log sequence to archive     0
Current log sequence              36
```

Step 3 : Enable flashback database on physical standby (if not enabled), by default flashback log will be written to the db_recovery_file_dest, they will enable the possibility to flashback the database to the point in time we will set in order to re-activate the physical standby once the test is finished.

To check whether flash back is on or not, use this query:

```
SQL> select flashback_on from v$database;
```

```
SQL> alter system set db_recovery_file_dest_size=2g
scope=both; System altered.
```

```
SQL> alter system set
db_recovery_file_dest='/u01/oracle/flash_recovery_ar
ea' scope=both;
System altered.
```

```
SQL> show parameters db_recovery
```

```
NAME TYPE VALUE -----
db_recovery_file_dest string
/u01/oracle/flash_recovery_area
db_recovery_file_dest_size big integer 2G
```

On the physical standby database we need to stop Redo Apply.

```
SQL> alter database recover managed standby database
cancel;
Database altered.
```

To enable flashback database the database needs to be shutdown cleanly, after that start up mount and enable flashback.

```
SQL> shutdown immediate;
SQL> startup nomount;
SQL> alter database mount standby database;
SQL> alter database flashback on;
Database altered.
```

Step 4 : Create a restore point on the standby database, a guaranteed restore point will make possible for us to flashback the database to this point in time, once our read write activities on its activated version will finish.

```
SQL> create restore point RESTORE_POINT_NAME
guarantee flashback database;
Restore point created.
```

Step 5 : Archive log current on primary database, We need to archive the current redo log on the primary database; it will be shipped to the standby server but not applied. This way we will assure that when flashing back and reactivating the standby we will have the archived logs up to the SCN of the restore point.

```
SQL> alter system archive log current;
System altered.
SQL> archive log list;
Database log mode                Archive Mode
```

```

Automatic archival          Enabled
Archive destination
USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence 35
Next log sequence to archive 37
Current log sequence        37

```

Step 6 : Stop redo log shipping to standby database, this step is performed on the Primary Database; we defer redo log shipping to the standby while it will be activated read write. First we check the actual values for the standby archive log destination.

```

SQL> show parameters log_archive_dest_state_2
NAME                                TYPE
VALUE
-----
log_archive_dest_state_2            string
ENABLE

```

```

SQL> show parameters log_archive_dest_2

NAME  TYPE  VALUE
-----
log_archive_dest_2 string SERVICE=DG2
VALID_FOR=(ALL_LOGFILES,PRIMARY_ROLE)
DB_UNIQUE_NAME=DG2 LGWR ASYNC
REOPEN=30

```

Once we are sure which is the destination to defer we can execute the defer command

```

SQL> ALTER SYSTEM SET
LOG_ARCHIVE_DEST_STATE_2=DEFER;
System altered.

SQL> show parameters log_archive_dest_state_2

NAME                                TYPE
VALUE
-----
log_archive_dest_state_2            string
DEFER

```

Step 7 : Activate standby database, at this moment the physical standby has stopped redo apply, we have enable flashback database and we have created a restore point. We are ready to activate the database and open it for read/write operations.

```
SQL> ALTER DATABASE ACTIVATE STANDBY DATABASE;
Database altered.
```

Step 8 : Startup mount force standby database

```
SQL> STARTUP MOUNT FORCE;
ORACLE instance started.
Total System Global Area 3154116608 bytes
Fixed Size                2032224 bytes
Variable Size             637541792 bytes
Database Buffers         2499805184 bytes
Redo Buffers              14737408 bytes
Database mounted.
```

Step 9 : Set standby database protection mode into maximum performance, we need to downgrade the protection mode to maximum performance because this mode permits to work without a standby database protecting the activated standby once opened.

When later we will flash back to the restore point, the physical standby automatically will get the protection mode defined on the primary.

```
SQL> alter database set standby database to maximize
performance;
Database altered.
```

Step 10 : Open the activated standby database

```
SQL> ALTER DATABASE OPEN;
Database altered.
```

Step 11 : Stop redo log shipping to the standby database, this step is performed on the Standby Database; we defer redo log shipping to the primary while it will be activated read write. First we check the actual values for the standby archive log destination

```
SQL> show parameters log_archive_dest_state_2
NAME                                     TYPE
VALUE
-----
log_archive_dest_state_2                string
ENABLE
```

```
SQL> show parameters log_archive_dest_2
NAME  TYPE  VALUE
```

```

-----
-----
log_archive_dest_2 string SERVICE=DG
VALID_FOR=(ALL_LOGFILES,PRIMARY
ROLE)
DB_UNIQUE_NAME=DG LGWR ASYNC
REOPEN=30

```

Once we are sure which is the destination to defer we can execute the defer command.

```

SQL> ALTER SYSTEM SET
LOG_ARCHIVE_DEST_STATE_2=DEFER;
System altered.

```

```

SQL> show parameters log_archive_dest_state_2

```

NAME	TYPE
log_archive_dest_state_2	string
DEFER	

Step 12 : The activated standby database now ready to be used (select, insert, update, delete)

Step 13 : Revert the activated standby database, on this step we will revert the activated database to its standby role, note that all information stored during the time the database was open read/write will be lost. You may want to make a backup if you need this information for later use.

Step 14 : Startup mount force the activated standby database, note the log sequences we have on the standby, we are as a matter of fact on a new incarnation

```

SQL> startup mount force;
ORACLE instance started.

```

```

Total System Global Area 3154116608 bytes
Fixed Size                2032224 bytes
Variable Size             637541792 bytes
Database Buffers         2499805184 bytes
Redo Buffers              14737408 bytes
Database mounted.

```

```

SQL> archive log list
Database log mode          Archive Mode
Automatic archival        Enabled
Archive destination
USE_DB_RECOVERY_FILE_DEST

```

```
Oldest online log sequence      1
Next log sequence to archive    2
Current log sequence            2
```

Step 15 : Flashback the activated standby database to the created restore point, on this step we are using the restore point `Before_App_Test` we created on step 5. We may have used also the SCN or a point in time.

```
SQL> flashback database to restore point
RESTORE_POINT_NAME;
flashback complete.
```

Step 16 : Convert the activated standby database to physical standby

```
SQL> alter database convert to physical standby;
Database altered.
```

Step 17 : Startup mount force the converted physical standby database, after having converted the database to the physical standby role we need to restart the database, notice that still at this point the sequences are after the restore point

```
SQL> STARTUP MOUNT FORCE;
ORACLE instance started.
```

```
Total System Global Area 3154116608 bytes
Fixed Size                  2032224 bytes
Variable Size               637541792 bytes
Database Buffers           2499805184 bytes
Redo Buffers                 14737408 bytes
Database mounted.
```

```
SQL> archive log list;
Database log mode           Archive Mode
Automatic archival         Enabled
Archive destination
USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence 1
Next log sequence to archive 2
Current log sequence        2
```

Step 18 : Start the redo apply in the standby database, which will start to apply archived logs from the primary that will take a minute until redo shipping is enabled on the primary.

```
SQL> alter database recover managed standby database
using current logfile disconnect from session;
Database altered.
```

Step 19 : Enable redo log shipping at the standby database, this is for switchover purpose.

```
SQL> alter system set
log_archive_dest_state_2=enable;
```

```
System altered.
```

```
SQL> select database_role, open_mode from
v$database;
```

```
DATABASE_ROLE      OPEN_MODE
-----
PHYSICAL STANDBY MOUNTED
```

Step 20 : Enable redo log shipping on the primary database we enable log shipping to the standby, when the stream of changes get to the standby its online logs will be cleared from the sequences stored on them when it was activated and open on read write mode

```
SQL> alter system set
log_archive_dest_state_2=enable;
System altered.
```

```
SQL> select database_role, open_mode from
v$database;
```

```
DATABASE_ROLE      OPEN_MODE
-----
PRIMARY            READ WRITE
```

```
SQL> alter system switch logfile;
System altered.
```

```
SQL> archive log list;
Database log mode           Archive Mode
Automatic archival         Enabled
Archive destination
USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence 36
Next log sequence to archive 38
Current log sequence       38
```

Step 21 : Check that archive logs are applied to the standby database

```
SQL> archive log list
Database log mode           Archive Mode
Automatic archival         Enabled
Archive destination
USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence 37
Next log sequence to archive 0
Current log sequence       38
```