Creating a Standby Database with RMAN

Contents

[Introduction 3](#_Toc458519607)

[Terminology 3](#_Toc458519608)

[Prepare the Primary Database 4](#_Toc458519609)

[Start the Primary database 4](#_Toc458519610)

[Enable Force Logging 4](#_Toc458519611)

[Create Standby Database's Pfile 4](#_Toc458519612)

[Update Tnsnames.ora 4](#_Toc458519613)

[Add Standby Redo Logs to Primary Database 4](#_Toc458519614)

[Prepare Primary Database to Send/Receive Redo Files 6](#_Toc458519615)

[Enable Archive Logging and Flashback 7](#_Toc458519616)

[Prepare the Standby Server 8](#_Toc458519617)

[Create a Password File for the Standby Database 8](#_Toc458519618)

[Create a Service for the Standby Database 8](#_Toc458519619)

[Create a Pfile for the Standby Database 8](#_Toc458519620)

[Create the Standby Structure 8](#_Toc458519621)

[Update the Standby Listener 8](#_Toc458519622)

[Start the Standby Instance 8](#_Toc458519623)

[Create the Standby Database 9](#_Toc458519624)

[Bugs & Foibles 10](#_Toc458519625)

[Post Clone Checks 10](#_Toc458519626)

[Start Managed Recovery 11](#_Toc458519627)

[Checking Managed Recovery 11](#_Toc458519628)

[Am I the Primary or Standby? 11](#_Toc458519629)

[Is There an Apply Gap? 11](#_Toc458519630)

[Status of the last 10 archived logs? 13](#_Toc458519631)

[What Point in Time Are the databases At? 13](#_Toc458519632)

[Stopping Managed Recovery 14](#_Toc458519633)

[Create an Application Service 14](#_Toc458519634)

[Amend Tnsnames.ora 14](#_Toc458519635)

[Create a Service on the Current primary 15](#_Toc458519636)

[Start the Service on the Current Primary 15](#_Toc458519637)

[Ensure That the Service Only Runs on the Primary 16](#_Toc458519638)

[Update RMAN Configuration for Primary & Standby Databases 17](#_Toc458519639)

[Configure Data Guard 17](#_Toc458519640)

[Configure the Listener 18](#_Toc458519641)

[Tnsnames.ora 19](#_Toc458519642)

[Start the DG Broker 19](#_Toc458519643)

[Start DGMGRL 19](#_Toc458519644)

[Create a DG Configuration 19](#_Toc458519645)

[Add a Standby Database 19](#_Toc458519646)

[Display Current Configuration 19](#_Toc458519647)

[Display Brief Primary Database Details 19](#_Toc458519648)

[Display Full Primary Database Details 20](#_Toc458519649)

[Display Brief Standby Database Details 21](#_Toc458519650)

[Display Full Standby Database Details 21](#_Toc458519651)

[Enable the Configuration 22](#_Toc458519652)

[Did it Work? 22](#_Toc458519653)

[Stopping the DG Broker 22](#_Toc458519654)

[Test Switchover 22](#_Toc458519655)

[Pre-Check Both Databases 23](#_Toc458519656)

[Switch Over 23](#_Toc458519657)

[Check for Correct Switch Over 23](#_Toc458519658)

[Switch Back 24](#_Toc458519659)

[Disabling the Standby Database 24](#_Toc458519660)

[Data Guard Troubleshooting 24](#_Toc458519661)

[Switch over 24](#_Toc458519662)

[Log Files 24](#_Toc458519663)

[What's Happening? 24](#_Toc458519664)

[Switch Over Works, One Way Only 25](#_Toc458519665)

# Introduction

The following outlines the steps followed in order to clone, using RMAN active duplicate, the AZDBA01 database, to create a standby database for use in a Data Guard environment on the Azure servers.

In this example, AZDBA01 will be cloned to a new database, AZDBA91, but the clone will have the same DB\_NAME initialisation parameter and DBID as the AZDBA01 database. This is standard for standby databases.

**Note:** To identify which database your session may be connected to, do not use NAME from V$DATABASE, use DB\_UNIQUE\_NAME instead. The following shows that we are connected to the standby database, AZDBA91 and that it has the same NAME as the primary, AZDBA01:

select name, db\_unique\_name from v$database;

NAME DB\_UNIQUE\_NAME

-------- -----------------

AZDBA01 AZDBA91

In the following example, the primary and standby remain on the same server, ORCDEVORC01, but in normal circumstances, this need not be the case, and indeed, is frowned upon. In addition, both databases are sharing the same disc, the G:\ drive.

# Terminology

* Primary database. The primary database - the one being cloned from. In RMAN, will be referred to as the ACTIVE DATABASE or the TARGET.
* Primary server. Where the primary database runs.
* Standby database. The standby database. The one being created by the clone process. In RMAN this is referred to as the CLONE or the AUXILIARY DATABASE.
* Standby server. Where the standby database runs.

# Prepare the Primary Database

## Amend the Primary Database Service

The primary could run as a standby. If it starts the service automatically, then the database will OPEN, regardless of its role, so the services have to be configured not to start automatically. For the most efficient shutdowns, immediate is advised.

oradim –edit –sid <primary\_database> -startmode manual –shutmode immediate

## Amend the Primary Database Listener

The database will not register with the listener when it runs as a standby. To ensure that the listener knows about the primary, when running as a standby, we need a static identifier in the listener.ora:

(SID\_DESC =

(SID\_NAME = AZDBA01)

(ORACLE\_HOME = C:\OracleDatabase\product\11.2.0\dbhome\_1)

)

The listener should be stopped and restarted:

lsnrctl stop

lsnrctl start

## Start the Primary database

Start, or ensure that the primary database was started, using an spfile. This can be checked by:

sqlplus / as sysdba

show parameter spfile

There *must* be a valid spfile name returned. If not, the database must be restarted using an spfile. You may have to create one from the current pfile, then restart. If necessary:

create spfile='?\database\spfile<primary\_database>.ora' from pfile='?\database\init<primary\_database>.ora';

shutdown

startup

In the above, the '?' is valid and is shorthand for %ORACLE\_HOME%.

## Enable Force Logging

Ensure that the primary database is running with force logging enabled. It must be MOUNTed or OPEN to allow this. There may be a delay while this takes effect as all currently unlogged changes will be required to complete.

select force\_logging from v$database;

Should return 'YES' if force logging is in force, if not, then run the following:

alter database force logging;

select force\_logging from v$database;

## Update Tnsnames.ora

Make sure that the tnsnames.ora has an entry for the new standby database. This can be checked by running:

tnsping standby\_database

There should be an ok status returned at the end of the output. If not, add the standby database to the primary server's tnsnames.ora file. If it hangs, make sure that there is actually a listener configured and running on the standby server.

## Add Standby Redo Logs to Primary Database

Add standby redo logs to the primary database. These will be used to allow the current primary database to be run as a standby database should the need ever occur to carry out a switchover.

There are required to be one standby log file group, with members, for each existing log file group on the primary. There is also a requirement for one extra standby log file group for each row in the V$THREAD view.

select count(\*) from v$thread;

Are there any existing standby logs?

select distinct type from v$logfile;

If the response is 'ONLINE' then proceed, otherwise, drop any existing standby logs:

select distinct 'alter database drop logfile group ' || to\_char(group#)

from v$logfile

where type = 'STANDBY'

order by 1;

The output from the above can be copied and pasted to remove the unwanted standby logs.

The (new) standby log file groups can be created with the output from the following command:

set serveroutput on size unlimited

declare

**-- Gap between top ONLINE and bottom STANDBY group#.**

**-- CHANGE THIS to suit your requirements.**

vDesiredOffset constant number := 10;

**-- Current highest and lowest ONLINE group.**

vMaxOnlineGroup v$logfile.group#%type;

vMinOnlineGroup v$logfile.group#%type;

**-- Current number of threads;**

vThreadCount number;

**-- New desired GROUP# for the STANDBY logs**

vNewGroup v$logfile.group#%type;

**-- How big is a log file?**

vMaxBytes v$log.bytes%type;

**-- PATH to the 'a' redo log.**

vRedoAPath v$logfile.member%type;

**-- PATH to the 'b' redo log.**

vRedoBPath v$logfile.member%type;

**-- Allows me to grab the members of the highest**

**-- ONLINE group of redo logs, to extract the paths.**

type tLogFileMembers is table of v$logfile.member%type

index by binary\_integer;

vLogFileMembers tLogFileMembers;

begin

**-- Get current maximum online group#.**

select min(group#), max(group#)

into vMinOnlineGroup, vMaxOnlineGroup

from v$logfile

where type = 'ONLINE';

**-- Get maximum size of a current logfile.**

select max(bytes)

into vMaxBytes

from v$log;

**-- Get the A and B paths. There could be more than 2 members.**

select member

bulk collect

into vLogFileMembers

from v$logfile

where group# = vMaxOnlineGroup;

**-- This assumes at least two members in each ONLINE group.**

**-- Any less might/will be a problem.**

vRedoAPath := substr(vLogFileMembers(1), 1, instr(vLogFileMembers(1), '\', -1));

vRedoBPath := substr(vLogFileMembers(2), 1, instr(vLogFileMembers(2), '\', -1));

**-- Get the thread count.**

select count(\*)

into vThreadCount

from v$thread;

**-- Build the desired standby groups.**

for onlineLog in (select distinct group# as gn

from v$logfile

where type = 'ONLINE'

order by 1)

loop

**-- If current max is 13, we want the minimum standby group to**

**-- be 23 + desired offset + 1. The minimum new group will be**

**-- that number.**

vNewGroup := onlineLog.gn + vMaxOnlineGroup - vMinOnlineGroup + vDesiredOffset + 1;

dbms\_output.put('alter database add standby logfile group ');

dbms\_output.put\_line(to\_char(vNewGroup) || ' (');

dbms\_output.put\_line('''' || vRedoAPath || 'stby' || to\_char(vNewGroup) || 'a.log'',');

dbms\_output.put\_line('''' || vRedoBPath || 'stby' || to\_char(vNewGroup) || 'b.log''');

dbms\_output.put\_line(') size ' || to\_char(vMaxBytes) || ';');

dbms\_output.put\_line(' ');

end loop;

**-- We also need an extra standby for each entry in V$THREAD.**

dbms\_output.put\_line('-- We also need one extra standby for each entry in V$THREAD.');

for extraLog in 1..vThreadCount

loop

vNewGroup := vNewGroup + 1;

dbms\_output.put('alter database add standby logfile group ');

dbms\_output.put\_line (to\_char(vNewGroup) || ' (');

dbms\_output.put\_line ('''' || vRedoAPath || 'stby' || to\_char(vNewGroup) || 'a.log'',');

dbms\_output.put\_line ('''' || vRedoBPath || 'stby' || to\_char(vNewGroup) || 'b.log'') ');

dbms\_output.put\_line('size ' || to\_char(vMaxBytes) || ';');

end loop;

end;

/

The output will resemble the following (abridged) and should be copied and executed to create the desired standby logfiles.

alter database add standby logfile group 24 (

'g:\MNT\ORADATA\azdba01\stby24a.log',

'g:\MNT\FAST\_RECOVERY\_AREA\azdba01\stby24b.log'

) size 104857600;

...

-- We also need one extra standby for each entry in V$THREAD.

alter database add standby logfile group 34 (

'g:\MNT\ORADATA\azdba01\stby34a.log',

'g:\MNT\FAST\_RECOVERY\_AREA\azdba01\stby34b.log'

) size 104857600;

The group numbers leave a gap of 10 entries between the current maximum online group number and the new lowest standby group number. This will allow new online groups to be added if required with continuing sequence numbers.

**Make sure that there are no existing STANDBY logfiles. If there are, you may need to adjust** vDesiredOffset **in the code above to skip over those. Alternatively, drop them.**

The size of the files in each standby log file group must be large enough to receive any log file on the primary database, so they must be sized according to the current maximum log file size.

The script also generates an additional redo log group addition, which is what is normally required. However, if there are more than one row in V$THREAD, then there needs to be one additional log group for each thread in V$THREAD

The script above may/has generated the following extra log group:

-- We also need one extra standby for each entry in V$THREAD.

alter database add standby logfile group 34 (

'g:\MNT\ORADATA\azdba01\stby34a.log',

'g:\MNT\FAST\_RECOVERY\_AREA\azdba01\stby34b.log'

) size 104857600;

## Prepare Primary Database to Send/Receive Redo Files

Although we are configuring AZDBA01 as the primary database, it can and may be used as a standby database, so it has to be able to *receive* redo log files. This configuration will be carried over to the standby database when it is cloned.

The primary database will have its db\_unique\_name and db\_name parameters set the same. On the standby, they will differ.

The following parameters will be in used when the database is running as a primary database.

alter system set db\_unique\_name='AZDBA01' scope=spfile;

alter system set log\_archive\_config='DG\_CONFIG=(AZDBA01,AZDBA91)' scope=spfile;

alter system set log\_archive\_dest\_1='location=use\_db\_recovery\_file\_dest' scope=spfile;

alter system set log\_archive\_dest\_2='service=azdba91 async valid\_for=(online\_logfiles,primary\_role) db\_unique\_name=AZDBA91' scope=spfile;

alter system set log\_archive\_dest\_state\_1=enable scope=spfile;

alter system set log\_archive\_dest\_state\_2=enable scope=spfile;

alter system set remote\_login\_passwordfile=exclusive scope=spfile;

alter system set LOG\_ARCHIVE\_FORMAT='arc\_%s\_%r\_%t.arc' scope=spfile;

The following parameters will be in used when the database is running as a standby database.

alter system set fal\_server=AZDBA91 scope=spfile;

alter system set db\_file\_name\_convert=

'g:\mnt\oradata\azdba91',

'g:\mnt\oradata\azdba01',

'g:\mnt\fast\_recovery\_area\azdba91',

'g:\mnt\fast\_recovery\_area\azdba01' scope=spfile;

alter system set log\_file\_name\_convert=

'g:\mnt\fast\_recovery\_area\azdba91',

'g:\mnt\fast\_recovery\_area\azdba01' scope=spfile;

alter system set standby\_file\_management=auto scope=spfile;

## Enable Archive Logging and Flashback

To determine if the database is in archive log mode and/or flashback mode, the following SQL will suffice:

select log\_mode, flashback\_on from v$database;

LOG\_MODE FLASHBACK\_ON

------------ ------------------

ARCHIVELOG YES

If ARCHIVELOG and FLASHBACK are already enabled, shutdown and startup the database to enable the new parameters above.

shutdown

startup

If the database is not yet in ARCHIVELOG mode (FLASHBACK cannot therefore be enabled) then the database must be put into ARCHIVELOG and flashback mode as follows. Note that it must be in ARCHIVELOG mode *before* FLASHBACK can be enabled.

shutdown

startup mount

alter database archivelog;

alter database open

alter database flashback on;

If the database is in ARCHIVELOG but not FLASHBACK, then simply enable FLASHBACK:

alter database flashback on;

shutdown

startup

The primary database is now ready to be cloned as a standby.

You must have restarted the database before continuing. The newly added standby parameters will not take effect until you do. Also, log shipping etc will not work either.

# Prepare the Standby Server

## Create a Password File for the Standby Database

Copy the password file from %ORACLE\_HOME%\Database\pwd<primary\_database>.ora to the standby server's location. Rename the file to suit the unique name of the standby database - %ORACLE\_HOME%\Database\pwd<standby\_database>.ora.

## Create a Service for the Standby Database

Open a command session *as administrator* and set the oracle environment appropriately, then enter the following command, all on one line:

oradim -new -sid <standby\_database> -startmode manual –shutmode immediate

## Create a Pfile for the Standby Database

Create a pfile, in your own login location, named init<standby\_database>.ora - initAZDBA91.ora in our example - and add the following single line to it:

DB\_NAME=AZDBA91

The file created should be copied to %oracle\_home%\database on the standby server. Permissions are such on the servers that while we have the ability to copy files into %ORACLE\_HOME%, we don't always the ability to subsequently edit them.

## Create the Standby Structure

Create the folder structure required by the standby database. For example, run the following in a cmd session to easily create the full paths:

mkdir g:\mnt\oradata\AZDBA91

mkdir g:\mnt\fast\_recovery\_area\AZDBA91

## Update the Standby Listener

Add an entry to the listener.ora file on the standby server. There must be an explicit entry for the standby database as it cannot auto-register itself on startup, because we never get it past the nomount stage. The following was added for our example:

(SID\_DESC =

(SID\_NAME = AZDBA91)

(ORACLE\_HOME = C:\OracleDatabase\product\11.2.0\dbhome\_1)

)

The listener service will have to be restarted:

lsnrctl stop

lsntcrl start

If you are unable to do this from the command line then you can do it from the Component Services utility off of the start menu.

## Start the Standby Instance

Start the standby instance:

set oracle\_sid=azdba91

sqlplus / as sysdba

-- It may have been started by oradim above, so …

shutdown

startup nomount pfile='?\database\initazdba91.ora'

exit

## Update the Standby Tnsnames.ora

The tnsnames.ora file on the standby server must have an entry for the primary and the new standby databases added.

Test – you must be able to connect to the SYS user, as SYSDBA, from *both* servers to *both* databases.

### On Primary Server

ping standby\_server

sqlplus sys/<password>@<primary\_database> as sysdba

sqlplus sys/<password>@<standby\_database> as sysdba

### On Standby Server

ping primary\_server

sqlplus sys/<password>@<primary\_database> as sysdba

sqlplus sys/<password>@<standby\_database> as sysdba

## Create the Standby Database

Connect to RMAN using a password for both the target and auxiliary databases. There must also be a tnsnames.ora alias used for the auxiliary database. For best results, use one on both databases. You can connect from either the primary or the standby servers, it makes no difference.

rman target sys/password@AZDBA01 auxiliary sys/password@AZDBA91

Run the following command:

run {

allocate auxiliary channel x1 device type DISK;

allocate auxiliary channel x2 device type DISK;

allocate auxiliary channel x3 device type DISK;

allocate channel d1 device type DISK;

allocate channel d2 device type DISK;

allocate channel d3 device type DISK;

allocate channel d4 device type DISK;

allocate channel d5 device type DISK;

duplicate target database

for standby

from active database

dorecover

spfile

parameter\_value\_convert

'G:\mnt\oradata\AZDBA01',

'G:\mnt\oradata\AZDBA91',

'G:\mnt\fast\_recovery\_area\AZDBA01',

'G:\mnt\fast\_recovery\_area\AZDBA91'

set control\_files

'G:\mnt\oradata\AZDBA91\control01.ctl',

'G:\mnt\fast\_recovery\_area\AZDBA91\control02.ctl'

set db\_unique\_name 'AZDBA91'

set db\_file\_name\_convert

'G:\mnt\oradata\AZDBA01',

'G:\mnt\oradata\AZDBA91',

'G:\mnt\fast\_recovery\_area\AZDBA01',

'G:\mnt\fast\_recovery\_area\AZDBA91'

set fal\_server 'AZDBA01'

set instance\_name 'AZDBA91'

set service\_names 'AZDBA91'

set audit\_file\_dest 'C:\ORACLEDATABASE\ADMIN\AZDBA91\ADUMP'

set dispatchers '(PROTOCOL=TCP) (SERVICE=AZDBA91XDB)'

set db\_recovery\_file\_dest 'G:\mnt\fast\_recovery\_area'

set dg\_broker\_start=false

set log\_file\_name\_convert

'G:\mnt\oradata\AZDBA01',

'G:\mnt\oradata\AZDBA91',

'G:\mnt\fast\_recovery\_area\AZDBA01',

'G:\mnt\fast\_recovery\_area\AZDBA91'

set log\_archive\_dest\_1

'location=use\_db\_recovery\_file\_dest valid\_for=(all\_logfiles,all\_roles) db\_unique\_name=AZDBA91'

set log\_archive\_dest\_2

'service=azdba01 async valid\_for=(online\_logfiles,primary\_role) db\_unique\_name=AZDBA01'

nofilenamecheck

;

release channel x1;

release channel x2;

release channel x3;

release channel d1;

release channel d2;

release channel d3;

release channel d4;

release channel d5;

}

exit

**Warning:**

The NOFILENAMECHECK parameter is required *only* when the clone is to a standby database on a *different* server. If the clone is to the same server, the parameter *must be removed*.

The PARAMETER\_VALUE\_CONVERT is *supposed* to rename the settings for the control\_files etc, but appears not to work (at least on Azure). By specifying the control\_files parameter above, this problem was worked around.

It is possible, perhaps desirable, to increase the number of disk, but not auxiliary, channels as this aids in the parallelism of the clone process. However, don’t allocate too many or you may swamp the network reducing efficiency. Five disk channels would probably be about the maximum advised.

## Bugs & Foibles

During the creation of a standby in the above manner, you may find the following errors at the end:

RMAN-05535: WARNING: All redo log files were not defined properly.

ORACLE error from auxiliary database: ORA-01511: error in renaming log/data files

ORA-01275: Operation RENAME is not allowed if standby file management is automatic.

These can be ignored. It's an Oracle "feature". There is a workaround, but it is not necessary and may cause other problems in renaming files with the possibility of overwriting the primary database log files as a side effect, when running both databases on the same server - as we are here.

RMAN-04014: startup failed: ORA-16024: parameter LOG\_ARCHIVE\_DEST\_1 cannot be parsed

Check, carefully, your parameter setting. VALID\_FOR is one word with an underscore, not 2 separate words. There should be an '=' with no spaces around it and brackets around the options, as in, valid\_for=(yada,yada) and so on.

## Post Clone Checks

set oracle\_sid=azdba91

sqlplus / as sysdba

show parameter instance\_name

show parameter service\_names

show parameter audit\_file\_dest

show parameter dispatchers

show parameter db\_recovery\_file\_dest

The output from the above should show the standby database name as appropriate, and not the primary. If not, run the appropriate command(s) below and bounce the database:

alter system set instance\_name='AZDBA91' scope=spfile;

alter system set service\_names='AZDBA91' scope=spfile;

alter system set audit\_file\_dest = 'C:\ORACLEDATABASE\ADMIN\AZDBA91\ADUMP' scope = spfile;

alter system set dispatchers=

'(PROTOCOL=TCP) (SERVICE=AZDBA91XDB)' scope=spfile;

alter system set db\_recovery\_file\_dest='G:\mnt\fast\_recovery\_area' scope=spfile;

startup force mount;

show parameter instance\_name

show parameter service\_names

show parameter audit\_file\_dest

show parameter dispatchers

show parameter db\_recovery\_file\_dest

The output from the above should now show the standby database name as appropriate, and not the primary.

select d.name, d.db\_unique\_name, d.database\_role, i.instance\_name

from v$database d, v$instance i;

NAME DB\_UNIQUE\_NAME DATABASE\_ROLE INSTANCE\_NAME

--------- ------------------------------ ---------------- -------------

AZDBA01 AZDBA91 PHYSICAL STANDBY azdba91

# Start Managed Recovery

The database is not yet in managed recovery mode, so to start it off we need to run the following command:

alter database recover managed standby database

using current logfile disconnect;

Database altered.

The database should now begin to apply any updates from the primary. You can force a log switch, on the primary database, as follows:

alter system archive log current;

archive log list

Database log mode Archive Mode

Automatic archival Enabled

Archive destination USE\_DB\_RECOVERY\_FILE\_DEST

Oldest online log sequence 3321

Next log sequence to archive 3330

Current log sequence 3330

Then after a few seconds,

select gap\_status

from v$archive\_dest\_status

where dest\_id=2;

You should expect to see 'NO GAP' but if there's a delay, you may see 'RESOLVABLE GAP' if the standby hasn't quite caught up. Anything else should be investigated before continuing.

On the standby, running ARCHIVE LOG LIST is a quick check that all is well, the current log sequence should match that on the primary:

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 3330

## Checking Managed Recovery

### Am I the Primary or Standby?

Example on a primary database:

select database\_role from v$database;

DATABASE\_ROLE

-------------

PRIMARY

Example on a standby database:

DATABASE\_ROLE

----------------

PHYSICAL STANDBY

The result will indicate the current role of this particular instance.

### Is There an Apply Gap?

The easiest method is to use dgmgrl after Data Guard has been configured (see below) as this gives up to date details of any gaps etc:

set oracle\_sid=azdbaxx

dgmgrl /

show configuration

...

Databases:

azdba01 - Primary database

azdba91 - Physical standby database

...

show database 'azdba91'

Database - azdba91

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: 137.00 KByte/s

Real Time Query: OFF

Instance(s):

azdba91

Database Status:

SUCCESS

Until Data Guard is configured, and running, you must query the databases to find any gaps. To check on the Primary:

select Dest\_name, destination, archived\_seq#, applied\_seq#, error, db\_unique\_name, gap\_status

from v$archive\_dest\_status

where status <> 'INACTIVE'

and dest\_name = 'LOG\_ARCHIVE\_DEST\_2';

Where 'n' is the dest\_id, usually 2, of the database init.ora parameter that ships logs to the standby database.

DEST\_NAME ARCHIVED\_SEQ# APPLIED\_SEQ# ERROR GAP\_STATUS

----------------------------------------------------------------------

LOG\_ARCHIVE\_DEST\_2 15975 15974 NO GAP

"NO GAP" is what you are hoping to see.

Or, alternatively, you can check on the standby:

select Dest\_name, destination, archived\_seq#, applied\_seq#, error, db\_unique\_name, gap\_status

from v$archive\_dest\_status

where status <> 'INACTIVE'

and dest\_name = 'STANDBY\_ARCHIVE\_DEST';

DEST\_NAME ARCHIVED\_SEQ# APPLIED\_SEQ# ERROR GAP\_STATUS

----------------------------------------------------------------------

STANDBY\_ARCHIVE\_DEST 15975 15975 NO GAP

On the primary database there will *usually* be a gap of 1 between the ARCHIVED\_SEQ# and the APPLIED\_SEQ# even if the standby shows that both are the same, and match the primary's ARCHIVED\_SEQ#. You can check if you wish:

select sequence#, dest\_id, applied

from v$archived\_log

where sequence# >= 15974

and dest\_id = 2;

SEQUENCE# DEST\_ID APPLIED

---------- ---------- -------

15974 2 YES

15975 2 YES

As before, the dest\_id matches the LOG\_ARCHIVE\_DEST\_2 init.ora parameter and usually refers to the standby database service. Adjust the query as necessary.

You can see from the results that the log with sequence# of 15975 has indeed been applied at the standby site, even though the above query shows a "gap". ☺

It is therefore best to run this particular query against the v$archive\_dest\_status on the standby, for the most accurate results!

### Status of the last 10 archived logs?

Run on the Primary:

select sequence#, dest\_id, name, applied, backup\_count, status

from v$archived\_log

where sequence# > (select -10 + max(sequence#) from v$archived\_log)

and lower(name) = 'azdba91'

order by sequence#;

You should see that all of the listed logs are applied, with the exception of the final one. Anything else needs to be investigated. Obviously, the lower case standby database name should be specified according to what that particular standby is actually named.

SEQUENCE# DEST\_ID NAME APPLIED BACKUP\_COUNT STATUS

------------------------------------------------------------------------

15723 2 azdba91 YES 0 A

15724 2 azdba91 YES 0 A

15725 2 azdba91 YES 0 A

15726 2 azdba91 YES 0 A

15727 2 azdba91 YES 0 A

15728 2 azdba91 YES 0 A

15729 2 azdba91 YES 0 A

15730 2 azdba91 YES 0 A

15731 2 azdba91 YES 0 A

15732 2 azdba91 NO 0 A

### What Point in Time Are the databases At?

Run on the Primary:

select scn\_to\_timestamp(current\_scn) as PRIMARY\_SCN from v$database;

PRIMARY\_SCN

-------------------------------

20-JUN-16 01.55.32.000000000 PM

Run on the Standby:

select current\_scn from v$database;

CURRENT\_SCN

-----------

54374184

Then, run on the Primary:

select scn\_to\_timestamp(54374184) as STANDBY\_SCN from dual;

STANDBY\_SCN

-------------------------------

20-JUN-16 01.32.56.000000000 PM

This has to be run on the Primary as the open mode of the standby prevents you from using it. Bear in mind that the standby database may be some time behind the primary due to some logs not yet having been shipped to the standby database. It is unlikely that both timestamps will coincide.

# Stopping Managed Recovery

*If it is necessary* to stop managed recovery, proceed as follows:

Login to the standby database as SYSDBA, and:

alter database recover managed standby database cancel;

The database is no longer applying any updates from the primary.

If the stoppage is going to be for some considerable time, it may be wise to DEFER archive log shipping on the *primary database* too. The following assumes that log\_archive\_dest\_2 is the one pointing to the standby database service:

alter system set log\_archive\_dest\_state\_2=’DEFER’ scope = memory;

If, of course, you wish to prevent the deferral from being rescinded after a reboot of the primary database, you should replace ‘memory’ with ‘both’.

Don’t forget to enable the destination after you have finished working on the standby again!

# Create an Application Service

Now that a pair of databases are acting as one, the tnsnames.ora must be changed to ensure that the application/user's connections connect to whichever database is running as the primary, without having to make changes to the application or the users' tnsnames.ora file every time there is a switch over.

## Amend Tnsnames.ora

Add the following entry, or a similar one, to the tnsnames.ora on *both* servers, and in the centralised tnsnames.ora of one is in use. If not, then the users will need an updated tnsnames.ora, possibly.

AZDBA =

(DESCRIPTION =

(ADDRESS\_LIST =

(ADDRESS = (PROTOCOL = TCP)(HOST = primary\_server)(PORT = 1521))

(ADDRESS = (PROTOCOL = TCP)(HOST = standby\_server)(PORT = 1521))

)

(CONNECT\_DATA =

(SERVER = DEDICATED)

(SERVICE\_NAME = AZDBA\_SERVICE)

)

)

The alias has been configured to be the databases' name, without the numeric suffix. The service is this alias with a \_SERVICE suffix. This way, DBAs and Data Guard can still directly access either AZDBA01 or AZDBA91, regardless of whether or not they are in primary or standby configuration.

This tnsnames.ora alias points the AZDBA at two separate servers, the primary database server and the standby database server. However, instead of connecting via a SID\_NAME or a SERVICE\_NAME that is the same as the database name, AZDBA01, we *must* now connect via a SERVICE\_NAME that is *different* to the database name as the database name is still used as the service name for the existing direct connections to AZDBA01 and AZDBA91.

## Create a Service on the Current primary

Login to the primary database as SYSDBA and create a new service to match the one used in the tnsnames.ora file above.

First, check the current service\_names parameter:

show parameter service\_names

NAME TYPE VALUE

------------------------------------ ----------- -------

service\_names string AZDBA01

We can see that only the existing service name, corresponding to the database's DB\_UNIQUE\_NAME, exists at present.

Next, create the new service:

begin

dbms\_service.create\_service(service\_name => 'AZDBA\_SERVICE',

network\_name => 'AZDBA\_SERVICE');

end;

/

It is best to have both parameters the same and both should match that used above in the new tnsnames.ora entry. The service will also be created on the current standby database automatically.

If you check the service\_names parameter again, nothing will have changed. You need to start the service for any changes to take place.

## Start the Service on the Current Primary

The service should now be started. As before, this must be carried out on the primary database:

begin

dbms\_service.start\_service(service\_name => 'AZDBA\_SERVICE');

end;

/

Then check the service\_names parameter again to see which service names are currently in use by the primary database:

SQL> show parameter service\_names

NAME TYPE VALUE

------------------------------------ ----------- ----------------------

service\_names string AZDBA01, AZDBA\_SERVICE

The primary database is now running with two separate service names, AZDBA01 and AZDBA\_SERVICE. This is as desired.

## Ensure That the Service Only Runs on the Primary

The AZDBA\_SERVICE defined and created above must only ever be running on the primary database. This will ensure that all connections using the AZDBA alias, which connects via the AZDBA\_SERVICE, will connect to either AZDBA01 or AZDBA91 regardless of which one is running as the current primary database.

Create a new database trigger which will start or stop the service as necessary, depending on whether the database started as a primary or a standby. This must be executed as the SYS user:

create or replace trigger azdba\_service\_trigger

after startup on database

declare

v\_role V$DATABASE.DATABASE\_ROLE%TYPE;

begin

--===================================================

-- Make sure we only start the AZDBA\_SERVICE on this

-- database if it is running as the primary database.

--===================================================

select database\_role

into v\_role

from v$database;

if (v\_role = 'PRIMARY') then

dbms\_service.start\_service('AZDBA\_SERVICE');

else

dbms\_service.stop\_service('AZDBA\_SERVICE');

end if;

end;

/

The naming convention used above is simply to append \_TRIGGER to the service name we are wishing to maintain.

If the database is started as a primary, the AZDBA\_SERVICE will be started. Connections will automatically be directed to the running primary database.

If the database is started as a standby, then the service will not be started.

If both databases happen to be down, the service will not be started, and connections will fail in the normal manner.

# Update RMAN Configuration for Primary & Standby Databases

The primary and standby databases should have their archivelog deletion policy updated to "applied on all standby backed up 2 times to device type disk". This will apply regardless of which instance is running as the primary as both databases have the same DBID.

Non primary-standby pairs only have their archived logs deleted after two successful backups. With a primary-standby pair, it is necessary to ensure that the files have also been applied to the current standby.

There is no need to register the standby database with RMAN, as this will create an error as the primary database's DBID is already registered.

rman target sys/password@azdba01 catalog rman11g/password@azrmn01

If you run the following command, the current configuration will be displayed:

show ARCHIVELOG DELETION POLICY;

RMAN configuration parameters for database with db\_unique\_name AZDBA01 are:

CONFIGURE ARCHIVELOG DELETION POLICY TO BACKED UP 2 TIMES TO DISK;

We need to adjust the setting now that we have a standby.

CONFIGURE ARCHIVELOG DELETION POLICY TO APPLIED ON ALL STANDBY BACKED UP 2 TIMES TO DEVICE TYPE DISK;

old RMAN configuration parameters:

CONFIGURE ARCHIVELOG DELETION POLICY TO BACKED UP 2 TIMES TO DISK;

new RMAN configuration parameters:

CONFIGURE ARCHIVELOG DELETION POLICY TO APPLIED ON ALL STANDBY BACKED UP 2 TIMES TO DISK;

new RMAN configuration parameters are successfully stored

exit;

Obviously, 'disk' in the above could very well be 'tape' or 'sbt\_tape' according to what we are doing exactly to run our RMAN backups.

# Configure Data Guard

Once the standby has been shown to work correctly, and is shipping logs etc, it is now in a position to be used to wait for some natural disaster to occur which requires a *failover* or tested periodically by carrying out a *switchover*.

In the current state, the switchover operation will require a large number of checks and SQL commands to be carried out by the DBA before the database can be considered ready to be switched over.

If we configure Data Guard then switching over is a simple matter of:

* Running dgmgrl on either the primary or standby server.
* Logging in as the SYS user, with a password. You need a password to create configurations, switchover or failover.
* Executing the command switchover to azdba91;

Having said that, Oracle advise that a number of checks are still performed prior to switching over in order to reduce the possibility of failures, and to speed up the actual switchover itself. These are detailed in the [AZURE - Standby Switchover 11g Standard Operating Procedures](http://cfsbckspsapp01/sites/fundsolutionIT/DevOps/Shared%20Documents/DBA%20Documents/Oracle%20Upgrade/Normans%20DBA%20Documentation/AZURE%20-%20Standby%20Switchover%2011g%20Standard%20Operating%20Procedures.docx) document in TFS.

If the current primary database is completely lost, then the commands should instead be:

Failover to azdba91;

This latter case will usually require a rebuild of the old primary database afterwards, but this can be done from within dgmgrl as well, and uses flashback to bring the database back from the dead, using the reinstate database command.

## Configure the Listener

There must be a static listener connection set up on both servers. The name of the connection must be DB\_UNIQUE\_NAME and the GLOBAL\_NAME must be DB\_UNIQUE\_NAME\_DGMGRL. You must edit listener.ora, on both servers, and add an entry to each.

The entry on the primary server should be:

(SID\_DESC =

(SID\_NAME = AZDBA01)

(GLOBAL\_DBNAME = AZDBA01\_DGMGRL)

(ORACLE\_HOME = C:\OracleDatabase\product\11.2.0\dbhome\_1)

)

While that on the standby server will be as follows:

(SID\_DESC =

(SID\_NAME = AZDBA91)

(GLOBAL\_DBNAME = AZDBA91\_DGMGRL)

(ORACLE\_HOME = C:\OracleDatabase\product\11.2.0\dbhome\_1)

)

Obviously, the listener will need to be restarted after any changes. You should restart it using the Component Services applet in Control Panel.

Additionally, *if the listener is running on a port other than 1521*, then the following entries need to be added, first to the primary listener:

(SID\_DESC =

(SID\_NAME = AZDBA01)

(GLOBAL\_DBNAME = AZDBA01\_DGB)

(ORACLE\_HOME = C:\OracleDatabase\product\11.2.0\dbhome\_1)

)

And to the standby listener:

(SID\_DESC =

(SID\_NAME = AZDBA91)

(GLOBAL\_DBNAME = AZDBA91\_DGB)

(ORACLE\_HOME = C:\OracleDatabase\product\11.2.0\dbhome\_1)

)

The \_DGB entry is used by the Data Guard Broker Process, DMON, to check the heartbeat of the different nodes in the configuration.

## Tnsnames.ora

Nothing needs to be changed in tnsnames.ora. The static listener connections and global\_dbname settings are all that is required and these are set up in listener.ora as per the instructions above.

## Start the DG Broker

On *both* databases, start the data guard broker as follows:

alter system set dg\_broker\_start=true scope=both;

exit;

## Start DGMGRL

On the primary server, start the dgmgrl utility as follows, and connect to the SYS user:

dgmgrl sys/password

## Create a DG Configuration

DGMGRL> create configuration 'dgmgrl\_configuration' as

primary database is azdba01

connect identifier is azdba01;

Configuration "dgmgrl\_configuration" created with primary database "azdba01"

## Add a Standby Database

DGMGRL> add database azdba91 as

connect identifier is azdba91

maintained as physical;

Database "azdba91" added

## Display Current Configuration

DGMGRL> show configuration

Configuration - dgmgrl\_configuration

Protection Mode: MaxPerformance

Databases:

azdba01 - Primary database

azdba91 - Physical standby database

Fast-Start Failover: DISABLED

Configuration Status:

DISABLED

## Display Brief Primary Database Details

DGMGRL> show database azdba01

Database - azdba01

Role: PRIMARY

Intended State: OFFLINE

Instance(s):

azdba01

Database Status:

DISABLED

## Display Full Primary Database Details

DGMGRL> show database verbose azdba01

Database - azdba01

Role: PRIMARY

Intended State: OFFLINE

Instance(s):

azdba01

Properties:

DGConnectIdentifier = 'azdba01'

ObserverConnectIdentifier = ''

LogXptMode = 'ASYNC'

DelayMins = '0'

Binding = 'optional'

MaxFailure = '0'

MaxConnections = '1'

ReopenSecs = '300'

NetTimeout = '30'

RedoCompression = 'DISABLE'

LogShipping = 'ON'

PreferredApplyInstance = ''

ApplyInstanceTimeout = '0'

ApplyParallel = 'AUTO'

StandbyFileManagement = 'AUTO'

ArchiveLagTarget = '900'

LogArchiveMaxProcesses = '4'

LogArchiveMinSucceedDest = '1'

DbFileNameConvert = 'g:\mnt\oradata\azdba91, g:\mnt\oradata\azdba01,

g:\mnt\fast\_recovery\_area\azdba91,

g:\mnt\fast\_recovery\_area\azdba01'

LogFileNameConvert = 'g:\mnt\fast\_recovery\_area\azdba91,

g:\mnt\fast\_recovery\_area\azdba01'

FastStartFailoverTarget = ''

InconsistentProperties = '(monitor)'

InconsistentLogXptProps = '(monitor)'

SendQEntries = '(monitor)'

LogXptStatus = '(monitor)'

RecvQEntries = '(monitor)'

ApplyLagThreshold = '0'

TransportLagThreshold = '0'

TransportDisconnectedThreshold = '30'

SidName = 'azdba01'

StaticConnectIdentifier = '(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=ORCDEVORC01)(PORT=1521))(CONNECT\_DATA=(SERVICE\_NAME=AZDBA01\_DGMGRL)(INSTANCE\_NAME=AZDBA01)(SERVER=DEDICATED)))'

StandbyArchiveLocation = 'use\_db\_recovery\_file'

AlternateLocation = ''

LogArchiveTrace = '0'

LogArchiveFormat = '%t\_%s\_%r.arc'

TopWaitEvents = '(monitor)'

Database Status:

DISABLED

## Display Brief Standby Database Details

DGMGRL> show database azdba91

Database - azdba91

Role: PHYSICAL STANDBY

Intended State: OFFLINE

Transport Lag: (unknown)

Apply Lag: (unknown)

Apply Rate: (unknown)

Real Time Query: OFF

Instance(s):

azdba91

Database Status:

DISABLED

## Display Full Standby Database Details

DGMGRL> show database verbose 'azdba91'

Database - azdba91

Role: PHYSICAL STANDBY

Intended State: OFFLINE

Transport Lag: (unknown)

Apply Lag: (unknown)

Apply Rate: (unknown)

Real Time Query: OFF

Instance(s):

azdba91

Properties:

DGConnectIdentifier = 'azdba91'

ObserverConnectIdentifier = ''

LogXptMode = 'ASYNC'

DelayMins = '0'

Binding = 'OPTIONAL'

MaxFailure = '0'

MaxConnections = '1'

ReopenSecs = '300'

NetTimeout = '30'

RedoCompression = 'DISABLE'

LogShipping = 'ON'

PreferredApplyInstance = ''

ApplyInstanceTimeout = '0'

ApplyParallel = 'AUTO'

StandbyFileManagement = 'AUTO'

ArchiveLagTarget = '900'

LogArchiveMaxProcesses = '4'

LogArchiveMinSucceedDest = '1'

DbFileNameConvert = 'g:\mnt\oradata\AZDBA01, g:\mnt\oradata\AZDBA91,

g:\mnt\fast\_recovery\_area\AZDBA01,

g:\mnt\fast\_recovery\_area\AZDBA91'

LogFileNameConvert = 'g:\mnt\oradata\AZDBA01, g:\mnt\oradata\AZDBA91,

g:\mnt\fast\_recovery\_area\AZDBA01,

g:\mnt\fast\_recovery\_area\AZDBA91'

FastStartFailoverTarget = ''

InconsistentProperties = '(monitor)'

InconsistentLogXptProps = '(monitor)'

SendQEntries = '(monitor)'

LogXptStatus = '(monitor)'

RecvQEntries = '(monitor)'

ApplyLagThreshold = '0'

TransportLagThreshold = '0'

TransportDisconnectedThreshold = '30'

SidName = 'azdba91'

StaticConnectIdentifier = '(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=ORCDEVORC01)(PORT=1521))(CONNECT\_DATA=(SERVICE\_NAME=AZDBA91\_DGMGRL)(INSTANCE\_NAME=AZDBA91)(SERVER=DEDICATED)))'

StandbyArchiveLocation = 'USE\_DB\_RECOVERY\_FILE\_DEST'

AlternateLocation = ''

LogArchiveTrace = '0'

LogArchiveFormat = '%t\_%s\_%r.arc'

TopWaitEvents = '(monitor)'

Database Status:

DISABLED

## Enable the Configuration

DGMGRL> enable configuration

Enabled.

## Did it Work?

DGMGRL> show configuration

Configuration - dgmgrl\_configuration

Protection Mode: MaxPerformance

Databases:

azdba91 - Primary database

azdba01 - Physical standby database

Fast-Start Failover: DISABLED

Configuration Status:

SUCCESS

The configuration has been enabled, and is working correctly.

## Stopping the DG Broker

*If it should become necessary* to disable Data Guard, then:

on *both* databases, primary and standby, stop the data guard broker as follows:

alter system set dg\_broker\_start=false scope=both;

exit;

## Test Switchover

It is wise to test switchovers in both directions, to be certain that all is working correctly.

### Pre-Check Both Databases

In SQL\*Plus, check the primary database details as follows:

select name, db\_unique\_name,database\_role

from v$database;

NAME DB\_UNIQUE\_NAME DATABASE\_ROLE

--------- ------------------------------ -------------

AZDBA01 AZDBA01 PRIMARY

And on the standby database, we get the following output:

NAME DB\_UNIQUE\_NAME DATABASE\_ROLE

--------- ------------------------------ ----------------

AZDBA01 AZDBA91 PHYSICAL STANDBY

### Switch Over

In dgmgrl, on either server, switch over to the current standby database:

DGMGRL> show configuration

Configuration - dgmgrl\_configuration

Protection Mode: MaxPerformance

Databases:

azdba01 - Primary database

azdba91 - Physical standby database

...

DGMGRL> switchover to azdba91

Performing switchover NOW, please wait...

Operation requires a connection to instance "azdba91" on database "azdba91"

Connecting to instance "azdba91"...

Connected.

New primary database "azdba91" is opening...

Operation requires startup of instance "azdba01" on database "azdba01"

Starting instance "azdba01"...

ORACLE instance started.

Database mounted.

Switchover succeeded, new primary is "azdba91"

### Check for Correct Switch Over

In SQL\*Plus, check the primary database details as follows:

select name, db\_unique\_name,database\_role

from v$database;

NAME DB\_UNIQUE\_NAME DATABASE\_ROLE

--------- ------------------------------ -------------

AZDBA01 AZDBA01 PHYSICAL STANDBY

And on the standby database, we get the following output:

NAME DB\_UNIQUE\_NAME DATABASE\_ROLE

--------- ------------------------------ ----------------

AZDBA01 AZDBA91 PRIMARY

We can see that the roles have been correctly reversed, azdba91 is now running as the primary with azdba01 as the new standby database.

### Switch Back

You must test that the switchover works both ways. This will prevent problems where it is possible to switch from azdba01 to azdba91 but a configuration problem prevents switching from azdba91 to azdba01.

DGMGRL> switchover to azdba01

Performing switchover NOW, please wait...

New primary database "azdba01" is opening...

Operation requires startup of instance "azdba91" on database "azdba91"

Starting instance "azdba91"...

ORACLE instance started.

Database mounted.

Switchover succeeded, new primary is "azdba01"

# Disabling the Standby Database

In order to disable the standby database, you need to:

* Cancel managed recovery and DEFER the primary database’s log\_archive\_dest\_2 – see [Stopping Managed Recovery](#_Stopping_Managed_Recovery).
* Turn off Data Guard, if in use – see [Stopping the DG Broker](#_Stopping_the_DG).

# Data Guard Troubleshooting

### Switch over

It takes a few minutes to perform a switchover. Bringing up the standby to a mount state seems to be the longest part. Be aware that during the mounting, running show configuration in a separate dgmgrl session will show spurious errors regarding log archive transport and/or mismatched DB\_UNIQUE\_NAME. Just wait and it should come up happily.

### Log Files

The log files for each database in a Data Guard configuration can be found in %ORACLE\_HOME%\diag\rdbms\<database>/<database>/trace/drc<database>.log which equates to the following for out two example databases above:

* %ORACLE\_HOME%\diag\rdbms\azdba01/azdba01/trace/drcazdba01.log
* %ORACLE\_HOME%\diag\rdbms\azdba91/azdba91/trace/drcazdba91.log

### What's Happening?

On the standby database, run this command:

select process,status,thread#,sequence#,block#,blocks

from v$managed\_standby;

The output will resemble this:

PROCESS STATUS THREAD# SEQUENCE# BLOCK# BLOCKS

--------- ------------ ---------- ---------- ---------- ----------

ARCH CLOSING 1 15933 2048 2004

ARCH CONNECTED 0 0 0 0

ARCH CLOSING 1 15936 1 561

ARCH CLOSING 1 15934 1 953

RFS IDLE 0 0 0 0

RFS IDLE 1 15937 315 1

RFS IDLE 0 0 0 0

MRP0 APPLYING\_LOG 1 15937 314 204800

The MRPn process is the one that carries out managed recovery. In the above, it is applying log sequence 15937. MRP0 in use means that the recovery was initiated with "alter database …. disconnect from session' as above. If the process is called MR(fg) then no disconnect was requested.

The RFS processes fetch logs from the primary as and when required. They are idle here, so nothing is coming across at the moment.

### Switch Over Works, One Way Only

If you find that you can switchover to one database ok, but cannot switch back, check the configuration and the database parameters:

instance\_name

db\_unique\_name

log\_archive\_dest\_2

service\_names

dispatcher

(audit\_file\_dest)

The listener.log may also be of use, as it could be showing the following:

TNS-12514: TNS:listener does not currently know of service requested in connect descriptor

22-JUN-2016 11:37:17 \* (CONNECT\_DATA=(SERVER=DEDICATED)(SERVICE\_NAME=AZDBA91)(CID=(PROGRAM=c:\oracledatabase\product\11.2.0\dbhome\_1\bin\ORACLE.EXE)(HOST=ORCDEVORC01)(USER=SYSTEM))) \* (ADDRESS=(PROTOCOL=tcp)(HOST=172.21.42.11)(PORT=64261)) \* establish \* AZDBA91 \* 12514

These messages will be shown if the standby database did not start for some reason, because the primary is still trying to connect to it regardless. Check database AZDBA91, in this case, to see what the SERVICE\_NAMES parameter is set to. You may have to run the following to correct things:

sqlplus / as sysdba

startup nomount (if the database didn't start)

show parameter service\_names

...

alter system set service\_names='AZDBA91' scope=both;

alter database mount;

Next, update the StaticConnectIdentifier in dgmgrl:

DGMGRL> edit database azdba91 set property StaticConnectIdentifier='(DESCRIPTION

=(ADDRESS=(PROTOCOL=tcp)(HOST=ORCDEVORC01)(PORT=1521))(CONNECT\_DATA=(SERVICE\_NAM

E=AZDBA91\_DGMGRL)(INSTANCE\_NAME=AZDBA91)(SERVER=DEDICATED)))';

Property "staticconnectidentifier" updated

The standby should now begin applying logs etc, however, your dgmgrl session is probably hung. You can leave it for a while (10 minutes should suffice) to see if it returns, or CTRL-C your way out if necessary.