Quick Guide to configuring

Oracle 11gR2 Data Guard

Physical Standby

Part I

Author: Jed Walker is the Manager of Database Operations for the Comcast Media Center in Centennial Colorado. He has been working with Oracle Database since 1997 and is an Oracle Certified Professional for 9i, 10g, and 11g Database.

Table of Contents

Table of Contents

Introduction

Data Guard is an Oracle feature that primarily provides database redundancy. This is done by having a standby (physical copy) database, preferably in another location and on separate disk. This standby database is maintained by applying the changes from the primary database to it. Standby databases can be maintained with either Redo (Physical standby) or SQL (Logical standby).

My intention with this paper is primarily to show that configuring Data Guard is not complex, and does not require any special skills or training to accomplish. It will step the reader quickly through the process of configuring a Physical Standby database for database redundancy of the primary system. My hope is that if you're new to Data Guard, just considering it, or worried that it is too hard to setup, this paper will help you quickly through the process and get you up and running.

If you find mistakes, additional important notes or considerations, please let me know.

Why Use Data Guard?

Each Oracle High Availability tool has its purpose. The reasons we are used Data Guard for the system that prompted this paper, are:

- Full database redundancy
- Fast recovery in the case of a failure
- Ability for clients to automatically reconnect after a failure
- Ability to offload the backup process to another server
- Provides very good Mean Time To Repair
- Not overly complex

System Setup

For this particular example, I will detail the environment a bit so you know what I'm working with. After writing the paper I used DBUA to create a brand-new database called JED and re-ran these steps to ensure that on a basic Oracle 11g database this would work. The Primary database is called JED and resides on a server called dev-db1. The physical standby will be called JED2 and reside on a server called dev-db2.

The Basic Stuff, that I shouldn't have to mention

For any production database there are some basics you should have in place. One of these is to run in ARCHIVELOG mode. This should be an obvious configuration for a production database. If you aren't already in ARCHIVELOG mode then you either need to start doing some reading and planning now, or have a very VERY good reason (I'm not sure there is one, but there is always an exception to every rule). To place your database in ARCHIVELOG mode:

```
SQL> archive log list;
```

If you see

```
     Database log mode              Archive Mode
```

then you are fine, otherwise do the following:

```
SQL> shutdown immediate

SQL> startup mount

SQL> alter database archivelog;

SQL> alter database open;

SQL> archive log list;
```

You should see

```
     Database log mode              Archive Mode
```

Preparing the Primary for Data Guard

You must now do some preparation on your Primary database so that it is ready for Data Guard.

First, for a Physical Standby to be an exact copy it must receive redo for the changes made to the primary. With Oracle a database user can instruct the database to not log redo (e.g. the NOLOGGING clause). For a physical standby database this is a big problem, so you must make sure that redo is logged regardless of what a user tells the database to do. To do this you can turn on Forced Logging. Here is how:

```
SQL> alter database force logging;

SQL> select name, force_logging from v$database;
```

In the FORCE_LOGGING column you should now see YES.

Next we'll make sure that when we add or drop datafiles on our primary database, that those files are also added or dropped on the standby. This is done with:

```
SQL> alter system set standby_file_management = 'AUTO';
```

Once this is done, we need to make sure the primary database has Standby Log Files. Standby Log Files are used by a standby database to store the redo it receives from the primary database. I create these on the primary for two reasons, 1) it may become a standby later and would then need them, 2) when we create the standby they will be created as part of that process. Standby log files should be created the same size as the online logfiles. Preferably you should have as many, or more, standby logfile groups as online logfile groups. To easily differentiate them in the V$ views, I like to number my standby logfile groups so they are in a different range from online logfiles (e.g. online logfile groups would be 1-6, standby logfile groups would be 11-16). To create a standby logfile group:

```
alter database add standby logfile group 11
('/oradata/JED/g11m01.sdo','/oradata/JED/g11m02.sdo') size 524288000;
```

If you are not using SSL for Redo Transport (likely you aren't), then you will need to use Oracle password file authentication. For this you must create a password file and have REMOTE_LOGIN_PASSWORDFILE set to EXCLUSIVE or SHARED. It is likely this is already done so check first, but if not:

```
SQL> alter system set remote_login_passwordfile=exclusive scope=spfile;
```
(and bounce)

```
OS> orapwd password=<the sys password>
```

You should also make sure your primary database has the db_unique_name parameter set for consistency.

```
SQL> show parameter db_unique_name
```

If it isn't set correctly for some reason, you can change it via the ALTER SYSTEM command.

Use Flashback Database

The next thing I highly recommend is to setup Flashback Database. Flashback Database allows you to "flashback" your database to a previous point in time. This is very useful when a failover occurs because you can use it to flashback your old primary database and then activate it as a Standby. If you don't have flashback correctly configured across your Data Guard configuration then you have to rebuild the Standby which means copying all those datafiles again. In addition to this use, Flashback Database can you save you from having to restore/recover your database in some situations.

To setup Flashback Database you need to configure the Fast Recovery Area first. To do this set the DB_RECOVERY_FILE_DEST and DB_RECOVERY_FILE_DEST_SIZE parameters. To configure these:

```
SQL> alter system set db_recovery_file_dest='&your_FRA_base_directory';

SQL> alter system set db_recovery_file_dest_size=400g;
```

Now that you have the Fast Recovery Area configured you need to turn on Flashback Logging. To do this:

```
SQL> alter database flashback on;
```

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

Verify that the FLASHBACK_ON column is YES. If you got the error ” ORA-01153: an incompatible media recovery is active” then you are likely doing this on a standby and need to cancel managed recovery and try again. Don’t forget to restart managed recovery when you’re done.

DO NOT GET BURNED! I discovered after a failover, that turning flashback on via “alter database” does not propagate to your standby databases with the redo (like many other things). You MUST run the alter database command on each of your standby databases. If you haven’t done this you’ll find later that you can’t flashback after a failover and will have to completely rebuild your old primary to make it a standby.

Server SQL*NET Configuration

Before we create the standby database, we need to ensure the databases in our configuration will be able to talk to each other. This is required before creating the database if we want to use the RMAN “duplicate from active database” feature. To do this we configure both the listener and TNSNames. You can do these by hand or by using the network configurations tools. I’m old school and these files are not complicated, so I prefer to do this by hand.

First you need to configure your primary and standby database in your listener. The reason for this is that while the databases can self-register, if you want to create the standby using RMAN it will need to be in NOMOUNT initially. In NOMOUNT the database instance will not self-register with the listener, so you must tell the listener it is there. Another item to note is that you must use dedicated server to connect when it is in NOMOUNT.

The TNS Names file on each server will need to be configured so that the databases can locate each other using the Service Names configured in the LOG_ARCHIVE_DEST_N and FAL_SERVER parameters (we’ll get to this later). Configure your files in a similar fashion to these examples.

	Primary (dev-db1 – server for JED)	Standby (dev-db2 – server for JED2)
Listener	SID_LIST_LISTENER = (SID_LIST = (SID_DESC = (GLOBAL_DBNAME = JED) (ORACLE_HOME = /oracle/product/11.2.0) (SID_NAME = JED)))	SID_LIST_LISTENER = (SID_LIST = (SID_DESC = (GLOBAL_DBNAME = JED2) (ORACLE_HOME = /oracle/product/11.2.0) (SID_NAME = JED2)))
TNS Names	JED2 = (DESCRIPTION =	JED = (DESCRIPTION =

	(ADDRESS_LIST = (ADDRESS = (PROTOCOL = TCP)(HOST = dev-db2)(PORT = 1521))) (CONNECT_DATA = (SERVICE_NAME = JED2) (SERVER = DEDICATED)))	(ADDRESS_LIST = (ADDRESS = (PROTOCOL = TCP)(HOST = dev-db1)(PORT = 1521))) (CONNECT_DATA = (SERVICE_NAME = JED) (SERVER = DEDICATED)))
Testing	OS> lsnrctl reload OS> tnsping JED2 <should return OK>	OS> lsnrctl reload OS> tnsping JED <should return OK>

Configuring Redo Transport

Now that the databases have the ability to talk to each other, our next step is to ensure that Redo is archived and transported appropriately. We'll configure this on the existing primary and then modify the standby after it is created.

To enable archiving of redo on our primary we use the following command:

```
SQL> alter system set log_archive_dest_1 = 'location=use_db_recovery_file_dest
valid_for=(all_logfiles, all_roles) db_unique_name=JED';
```

This statement says to use the DB_RECOVERY_FILE_DEST as the location to archive to, that it should be used to archive all logfiles in any (all) roles, and that this is for the database JED. The manual says to use online_logfiles; however, this will result in the Standby being unable to archive the standby logfiles because they are not online logfiles. If you use all_logfiles both the primary and standby will be able to archive logfiles whether they are online or standby. You'll want this if you're backing up on the standby and want to have archived logfiles backed up there too. You could configure this to archive only on the primary or only on the standby if you'd like.

Next we configure the transport of the redo to the Standby database.

```
SQL> alter system set log_archive_dest_2 = 'service=JED2 async
valid_for=(online_logfile,primary_role) db_unique_name=JED2';
```

This statement says that if it is the primary database, it should transport online logfiles using the service name JED2 and that the target is the database JED2.

At this point, I should note that the STANDBY_ARCHIVE_DEST parameter is deprecated and not needed. While troubleshooting I had many well-intentioned suggestions to add this parameter, but about all you'll get from it is "ORA-32004: obsolete or deprecated parameter(s) specified for RDBMS instance" when you start your database.

The next parameter is FAL_SERVER. This specifies where the database should look for missing archive logs if there is a gap in the logs. It is used when the database is in the standby role and has a gap in the redo it has received. This type of situation occurs when redo transport is interrupted, for example when you do maintenance on the standby server. During that maintenance no logs would have been transported and a gap would exist. Setting this parameter allows the standby to find the missing redo and have it transported.

```
SQL> alter system set fal_server = 'JED2';
```

Note that the FAL_CLIENT parameter is deprecated in 11g and not needed.

We then need to let the database know what other databases are in the Data Guard configuration.

```
SQL> alter system set log_archive_config = 'dg_config=(JED,JED2)';
```

Once this is done, we are ready to prepare the environment for the standby and create the standby.

Preparing the Standby environment

We are now ready to setup the standby environment for creation of the Standby database. There are many different ways to perform many of these steps. What I am showing here is the way I found worked best for me. You should experiment with other options to see what works best for you.

First, we need to create a password file and server parameter file (spfile) for the new standby database. The password file can be copied directly over, and only needs it name changed. For example, on the primary the files is $ORACLE_HOME/dbs/orapwJED. We can copy it over to the standby server and place it in the same location, but with the standby SID instead of the primary SID, i.e. $ORACLE_HOME/dbs/orapwJED2.

To create the standby spfile, on the primary create an initialization parameter file (pfile):

```
SQL> create pfile from spfile;
```

Now, before I continue I want to discuss a seemingly nice feature – creation of the SPFILE via RMAN. This feature is designed to allow you to let RMAN create your spfile for you on your standby location. Here is why I don't use it.

1. I have to copy the password file over to the standby anyway, so it doesn't save me having to copy files over.
2. To use this feature you still have to do a lot of replacement work by setting the parameter_value_convert parameter and the SPFILE clause and the various SET statements to make sure everything is correct.

I find it easier to copy a pfile over (you can even just cut and paste), rename it, and then change the parameters in it. This is easy to do and you'll learn a lot by reviewing everything and making sure it is correct. I spend less time doing this and fixing things than going through the RMAN SPFILE creation. So, with that said…

After creating your primary pfile, copy it over to the standby server in the same location, but change the name to have the standby SID (e.g. initJED.ora will become initJED2.ora). You will now need to make the following changes to it (and note that in your situation there might be others, review your pfile carefully):

- Depending on your configuration and file locations on the standby you might needed to change the AUDIT_FILE_DEST, CONTROL_FILES and DISPATCHERS parameters (there may be others, but be careful, not everything should be changed this way).
- In LOG_ARCHIVE_DEST_1 change db_unique_name to the standby SID (in this case JED2).
- In LOG_ARCHIVE_DEST_2 change the service and db_unique_name to the primary SID (in this case JED).
- Change FAL_SERVER to the primary SID (JED).
- Add the following parameters:
    - db_unique_name=JED2
    - db_file_name_convert and log_file_name_convert if you are changing the location of the datafiles and/or logfiles. I like to do this to make it obvious whose datafiles they are (e.g. /oradata/JED versus /oradata/JED2 by using db_file_name_convert=/oradata/JED','/oradata/JED2')

Now, on your standby server create the directory structures and modify files to support the new database. These should at least be:

- $ORACLE_BASE/admin/$ORACLE_SID
- $ORACLE_BASE/admin/$ORACLE_SID/adump (or whatever your audit_file_dest is)
- Datafile directories
- Controlfile directories
- Logfile directories
- DB Recovery File Destination (Fast Recovery Area)
- Add your standby to the /etc/oratab file.

You should now be ready to startup your standby instance for creation of the database. You also want to make sure you create an spfile in the process. To do this:

```
SQL> startup nomount pfile=<ORACLE_HOME>/dbs/initJED2.ora

SQL> create spfile from pfile;

SQL> shutdown

SQL> startup nomount

SQL> show parameter spfile

SQL> exit
```

The “show parameter spfile” should show the location of your spfile, and your standby should be in NOMOUNT.

Creating the Standby Database

Similar to the previous section there are many different ways you can create the standby database. With 11g I use RMAN duplication to do the creation because it is easy. In the previous section we copied over the password file and the pfile, modified it, and created the spfile. This makes the creation command with RMAN simpler; however, you could also skip manual creation of the password and spfile and have RMAN do it using the SPFILE, PARAMETER_VALUE_CONVERT, and SET syntax.

The command to create the Standby with RMAN is quite simple. It instructs RMAN to perform a duplication to the Auxiliary (standby) database for creation of a standby by making the copy directly from the active database. This is nice because you do not need to have backups staged on the standby server. With storage technologies available today the copy can be done in much faster and sometimes simpler ways, but for the purpose of showing this 11g feature and because it is quite simple I like to use it when possible.

```
RMAN> connect target sys@JED

RMAN> connect catalog <catalogowner>@<catalogdb>

RMAN> connect auxiliary sys@JED2

RMAN> duplicate target database for standby from active database;
```

- As of 11.2.0.2.0 you can connect to the target with “connect target”; however, if you don’t specify the username a duplication to standby will fail later with “invalid username/password”.

While this command is running I like to tail the standby alert log and see what is going and watch for errors. Note that it is normal and OK to get “ORA-27037: unable to obtain file status” on the online and standby log files.

To perform the duplication in parallel to improve performance you can allocate primary and standby channels and then run the duplicate command.

```
run
{
 allocate channel chan1 type disk;
 allocate channel chan2 type disk;
 allocate channel chan3 type disk;
 allocate channel chan4 type disk;
 allocate auxiliary channel aux1 type disk;
 allocate auxiliary channel aux2 type disk;
 allocate auxiliary channel aux3 type disk;
 allocate auxiliary channel aux4 type disk;
 duplicate target database for standby from active database
;
}
```

If everything works you should see something similar to the following in RMAN:

```
Finished Duplicate Db at 07-MAY-10
```

It is at this point that I like to turn on flashback logging for the standby:

```
SQL> alter database flashback on;
```

Starting Redo Apply on the Standby

Starting and stopping Redo Apply on the standby is very easy. Oracle Data Guard provides the ability for the standby to apply the redo itself. To start apply:

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE USING CURRENT LOGFILE
DISCONNECT FROM SESSION;
```

This starts the recovery process using the standby logfiles that the primary is writing the redo to. It also tells the standby to return you to the SQL command line once the command is complete. If you want to stop the recovery you do:

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE CANCEL;
```

Verifying that Redo Apply is working

OK, so we'd like to think all we have to do is watch it work, but sometimes things aren't that perfect. You should now verify that the redo is being applied on your standby. First we want to make sure our destinations are all valid on both the primary and the standby:

```
SQL> select DEST_ID, STATUS, DESTINATION, ERROR from V$ARCHIVE_DEST where DEST_ID
<=2;
```

The destinations should show as VALID.

Now, we'll see if the redo is actually being applied. From the primary run:

```
SQL> select SEQUENCE#, FIRST_TIME, NEXT_TIME, APPLIED, ARCHIVED from
V$ARCHIVED_LOG where name = 'JED2' order by FIRST_TIME;
```

You should see YES for the APPLIED and ARCHIVED columns if both archiving and redo apply are working correctly. Many examples of this query use "order by SEQUENCE#", but I don't recommend this. If you order by SEQUENCE# and then do a failover the sequence numbers will start again at 1 and you won't see your latest records at the end of the result-set. The first time I did this I could not figure out why I wasn't getting any new rows for archived redo. In fact, I was, but I just wasn't seeing them. As a result, I always order this query by FIRST_TIME.

If you notice that logs aren't applying, it is possible you might have a gap in your redo, in which case the standby cannot apply. If you have the FAL_SERVER parameter set correctly, this shouldn't be a problem. You can check to see if there are any gaps in the redo, by running the following query on the primary:

```
SQL> select STATUS, GAP_STATUS from V$ARCHIVE_DEST_STATUS where DEST_ID = 2;
```

It should return VALID and NO GAP if everything is OK. If you want to play around with this and see how the FAL_SERVER works you can shutdown your standby, switch out several logs, wait a bit, bring your standby back up, and switch another log out. You should soon see a GAP appear. If your FAL_SERVER parameter is correct on the standby and points to the service name of the primary, then the logs should be brought over and applied.

The V$DATAGUARD_STATUS view is very useful for looking for errors or just seeing what has happened. You can query it on the primary and standby to see status for that database.

```
SQL> select * from V$DATAGUARD_STATUS order by TIMESTAMP;
```

All of this is good, but sometimes you want to really know the data is there. A more reassuring way to verify is to actually check the standby and verify that the new data is there. You can do this by changing the standby's role to readonly. First you'll need to stop managed recovery and then run:

```
SQL> ALTER DATABASE OPEN READ ONLY;
```

You can now run your query to see that the changes have come across. When you're done, do not forget to take your database back to MOUNT and restart recovery. In Part II, you'll find out how you can do this type of check more easily with Oracle 11g Active Data Guard.

Conclusion

At this point you now have a Data Guard Configuration working and therefore a redundant copy of your database available. I don't want to leave you without going through switchover, failover, rebuilding, and other items, but that will have to wait until Part II of this paper. Besides, if you got this done then I need some way to bring you back for more!

I hope this paper helps you get your Data Guard configuration set up easily and quickly.