Creating a Physical Standby database using RMAN restore database from service (Doc ID 2283978.1)

In this Document

APPLIES TO:

Oracle Database - Enterprise Edition - Version 12.1.0.2 and later
Oracle Cloud Infrastructure - Database Service - Version N/A and later
Oracle Cloud at Customer - Version N/A and later

Information in this document applies to any platform.

GOAL

NOTE: In the images and/or the document content below, the user information and data used represents fictitious data from the Oracle sample schema(s) or Public Documentation delivered with an Oracle database product. Any similarity to actual persons, living or dead, is purely coincidental and not intended in any manner.

Maximum Availability Architecture

The Maximum Availability Architecture (MAA) defines Oracle's most comprehensive architecture for reducing downtime for scheduled outages as well as preventing, detecting and recovering from unscheduled outages. Real Application Clusters (RAC) and Oracle Data Guard are integral components of the Database MAA reference architectures and solutions.

More detailed information, such as a discussion of the purpose of MAA and the benefits it provides, can be found on the Oracle Technology Network (OTN) at http://www.oracle.com/technetwork/database/features/availability/maa-096107.html

Purpose of this Document

Provide a step-by-step guide for instantiating a standby database using the RMAN "from service" clause to copy directly from the primary database through an Oracle Net connection.

NOTE:

- This document applies to Oracle Database Server versions 12.1 to 19c and higher.
- SECTION SIZE support is available. The section size clause used with multiple RMAN channels enables parallelization of individual files by dividing large files into smaller pieces. This improves the overall efficiency of parallelization across channels.
- Encryption is supported.
- Compression is supported. It is not recommended to use compression on backups or data that has already been compressed (e.g. using OLTP, HCC compression) or encrypted since the compression benefits is very small and the overall impact (e.g. CPU resources and increased elapsed time) can be significant.

About RMAN 'FROM SERVICE' clause

The RMAN 'from service' clause enables the restore and recover of primary database files to a standby database across the network. This functionality can be used to instantiate a standby database in lieu of the RMAN DUPLICATE DATABASE command and is more intuitive and less error prone thus saving time. Additionally, utilizing the SECTION SIZE clause with multiple RMAN channels improves the efficiency of parallelization of the restore, further improving instantiation times.

NOTE: This 'FROM SERVICE' method can be used to restore or recover an entire database, individual data files, control files, server parameter file, or tablespaces. This method is useful in synchronizing the primary and standby database.

This paper assumes that the following conditions exist:

1. The network between the primary and standby sites is reliable and has been assessed and determined to support the peak redo generation rate of the primary. See note 2275154.1 for details on assessing the network.
2. A primary database utilizing ASM for data file storage as well as Oracle Managed Files(OMF).
3. The primary database is in archive log mode.
4. The primary database online redo logs:
   1. are identical in size
   2. are sized so they do not switch more than 6 times per hour at peak redo generation (this can significantly impact redo apply in Data Guard)
   3. reside on the DATA disk group
   4. are multiplexed on NORMAL redundancy disk groups (HIGH redundancy disk groups are not multiplexed)
   5. have a minimum of 3 groups for each thread of a RAC database
5. Password and spfile are stored in ASM.
6. The target standby host has all the required Oracle software installed and configured.
7. The standby database database software matches the primary database software. Including PSU/RU/RUR and one off patches.
8. The standby target database storage will utilize ASM storage and OMF.

9. The standby target resides on separate hardware.
10. If role separation is used in your environment set the environment based on the roles with oracle or grid. In our example the oracle user owns both grid and database software installations.

All of the example names illustrated in this document use the following naming:

Hosts and Databases Used in this Example		
	Primary	**Standby**
Hosts	<primaryhost1>, <primaryhost2>	<standbyhost1>, <standbyhost2>
Database Unique Name	<primary unique name>	<standby unique name>
Instance names	<primary unique name>1, <primary unique name>2	<standby unique name>1, <standby unique name>2

SOLUTION

Steps to Create a Physical Standby Database using "RESTORE DATABASE FROM SERVICE"

The following are the steps used to create the Data Guard standby database:

PRIMARY DATABASE

1. ***Put primary database in forced logging mode***

   Database force logging is recommended so that all changes to the primary database are replicated to the standby database regardless of NOLOGGING settings. To enable force logging, use the following command on the primary:

```
[oracle@<primaryhost1>]$ sqlplus / as sysdba
SQL> alter database force logging;
SQL>exit
```

2. ***Create Standby Redo Logs***

   Standby Redo Logs enable real time redo apply where the redo is applied as it is received rather than when a complete archived log is received. This improves standby currency and reduces potential data loss.

   Per MAA best practices Standby Redo Logs(SRL) are recommended to:
   - be identical size as online redo logs (to the byte)
   - have groups assigned to a thread in RAC configurations
   - be single member groups
   - have the same number of groups per thread as online redo log groups
   - reside on DATA disk group

   Create standby redo logs on the primary database that are the exact same size as the online redo logs. Creating the SRLs on the primary before instantiation will ensure they are defined on the standby during instantiation process. It will also ensure that standby redo log files are available on the current primary when the current primary becomes a standby.

   Oracle recommends having the same number of standby redo log groups as there are online redo log groups for each thread. Our primary database has 3 online redo log groups per thread. We therefore need 3 Standby Redo Log files per thread for the standby. As per MAA Best Practice we recommend to create only one member for standby redo log. For example:

Check existing redo log members and their sizes.

```
[oracle@<primaryhost1>]$ sqlplus / as sysdba

SQL> select thread#,group#,bytes,status from v$log;

THREAD#    GROUP#     BYTES          STATUS

---------- ---------- -------------- ----------------
1          1          4294967296     CURRENT
1          2          4294967296     UNUSED
1          3          4294967296     UNUSED
2          4          4294967296     CURRENT
2          5          4294967296     UNUSED
2          6          4294967296     UNUSED
```

Create the Standby Redo Logs:

```
[oracle@<primaryhost1>]$ sqlplus / as sysdba

SQL> alter database add standby logfile thread 1
group 7 ('+DATAC1')  size 4294967296,
group 8 ('+DATAC1')  size 4294967296,
group 9 ('+DATAC1')  size 4294967296,

             group 10 ('+DATAC1')  size 4294967296;


SQL> alter database add standby logfile thread 2
group 11 ('+DATAC1')  size 4294967296,
group 12 ('+DATAC1')  size 4294967296,
group 13 ('+DATAC1')  size 4294967296,

            group 14 ('+DATAC1')  size 4294967296;
```

3. ***Enable Standby File Management***

On the primary database set STANDBY_FILE_MANAGEMENT=AUTO.

```
[oracle@<primaryhost1>]$ sqlplus / as sysdba
SQL> alter system set STANDBY_FILE_MANAGEMENT=AUTO scope=both sid='*';
SQL>exit
```

4. ***Password File Copy***

Copy the password file from the primary database to the first standby host.

```
[oracle@<primaryhost1>]$ srvctl config database -d as19_fra1gh | grep 'Password file'
Password file: +DATAC1/<primary unique name>/PASSWORD/passwd  <-- this file is what needs to
be copied to /tmp and scp'd to the standby (result may differ)

[oracle@<primaryhost1>]$ export ORACLE_SID=+ASM1
[oracle@<primaryhost1>]$ export ORACLE_HOME=/u01/app/12.2.0.1/grid
[oracle@<primaryhost1>]$ export PATH=$ORACLE_HOME/bin:$PATH

[oracle@<primaryhost1>]$asmcmd cp +DATAC1/<primary unique name>/PASSWORD/passwd /tmp/passwd.
<primary unique name>
copying +DATAC1/<primary unique name>/PASSWORD/passwd -> /tmp/passwd.<primary unique name>
```

```
[oracle@<primaryhost1>]$ scp /tmp/orapw<standby unique name>1
oracle@<standbyhost1>:/tmp/orapw<standby unique name>
```

NOTE: If Transparent Data Encryption (TDE) is enabled on the primary, the TDE wallet must be copied to the standby also.

5. ***Create a pfile from the primary database and scp to standby***

```
[oracle@<primaryhost1>]$ export ORACLE_HOME=/u01/app/oracle/product/12.2.0.1/dbhome_1
[oracle@<primaryhost1>]$ export ORACLE_SID=<primary unique name>1
[oracle@<primaryhost1>]$ export PATH=$ORACLE_HOME/bin:$PATH
[oracle@<primaryhost1>]$ sqlplus / as sysdba

SQL> create pfile='/tmp/primary.pfile' from spfile;

[oracle@<primaryhost1>]$ scp /tmp/primary.pfile oracle@<standbyhost1>:/tmp/standby.pfile
```

6. ***Create net alias' for the primary and standby databases***

TNS Aliases used in the Oracle Data Guard configuration

(All entries should exist in tnsnames.ora of all primary and standby instances)

```
<primary unique name> =
(DESCRIPTION =
  (ADDRESS_LIST =
    (ADDRESS=(PROTOCOL= TCP) (HOST=prmy-scan)(PORT=<PORT>))
  )
  (CONNECT_DATA =
    (SERVER = DEDICATED)
    (SERVICE_NAME =<primary database service name>)
  )
)


<standby unique name> =
(DESCRIPTION =
  (ADDRESS_LIST =
    (ADDRESS=(PROTOCOL = TCP) (HOST=stby-scan)(PORT=<PORT>))
  )
  (CONNECT_DATA =
    (SERVER = DEDICATED)
    (SERVICE_NAME =<standby database service name>)
  )
)
```

# RAC ONLY create an entry for each primary and standby instance #

```
<primary unique name>1 =
(DESCRIPTION =
  (ADDRESS_LIST =
    (ADDRESS=(PROTOCOL= TCP) (HOST=prmy-scan)(PORT=<PORT>))
  )
  (CONNECT_DATA =
    (SERVER = DEDICATED)
    (SERVICE_NAME =<primary database service name>)
    (INSTANCE_NAME=<primary instance 1 SID_NAME>)
  )
)

<primary unique name>2 =
(DESCRIPTION =
  (ADDRESS_LIST =
    (ADDRESS=(PROTOCOL= TCP) (HOST=prmy-scan)(PORT=<PORT>))
  )
  (CONNECT_DATA =
    (SERVER = DEDICATED)
    (SERVICE_NAME =<primary database service name>)
    (INSTANCE_NAME=<primary instance 2 SID_NAME>)
  )
)

<standby unique name>1 =
(DESCRIPTION =
  (ADDRESS_LIST =
    (ADDRESS=(PROTOCOL = TCP) (HOST=stby-scan)(PORT=<PORT>))
  )
  (CONNECT_DATA =
    (SERVER = DEDICATED)
    (SERVICE_NAME =<standby database service name>)
    (INSTANCE_NAME=<standby instance 1 SID_NAME>)
  )
)

<standby unique name>2 =
(DESCRIPTION =
  (ADDRESS_LIST =
     (ADDRESS=(PROTOCOL = TCP) (HOST=stby-scan)(PORT=<PORT>))
  )
  (CONNECT_DATA =
     (SERVER = DEDICATED)
     (SERVICE_NAME =<standby database service name>)
     (INSTANCE_NAME=<standby instance 2 SID_NAME>)
  )
)

# < create an entry for each instance of primary and standby if more than two > #

##### END RAC ONLY########
```

STANDBY DATABASE

7. ***Create Audit Directory***

   On all standby hosts create the audit directory for the standby database.

```
[oracle@<standbyhost1>]$ mkdir -p /u01/app/oracle/admin/<standby unique name>/adump
[oracle@<standbyhost2>]$ mkdir -p /u01/app/oracle/admin/<standby unique name>/adump
```

8. ***Register the standby database with clusterware***

   Register the standby with clusterware and start the database nomount

   Single instance example:

```
[oracle@<standbyhost1>]$ export ORACLE_HOME=/u01/app/oracle/product/12.2.0.1/dbhome_1
[oracle@<standbyhost1>]$ export ORACLE_SID=<standby unique name>1
[oracle@<standbyhost1>]$ export PATH=$ORACLE_HOME/bin:$PATH
```

```
[oracle@<standbyhost1>]$ srvctl add database -db <standby unique name> -oraclehome
/u01/app/oracle/product/12.2.0.1/dbhome_1 -dbtype SINGLE -instance <standby unique name>1 -
node <standbyhost1> -dbname <primary unique name> -diskgroup RECOC1,DATAC1 -role
physical_standby
```

RAC example.

```
[oracle@<standbyhost1>]$ export ORACLE_HOME=/u01/app/oracle/product/12.2.0.1/dbhome_1
[oracle@<standbyhost1>]$ export ORACLE_SID=<standby unique name>1
[oracle@<standbyhost1>]$ export PATH=$ORACLE_HOME/bin:$PATH

[oracle@<standbyhost1>]$ srvctl add database -db <standby unique name> -oraclehome
/u01/app/oracle/product/12.2.0.1/dbhome_1 -dbtype RAC -dbname <db_name parameter> -diskgroup
RECOC1,DATAC1 -role physical_standby

[oracle@<standbyhost1>]$ srvctl add instance -database <standby unique name> -instance
<standby unique name>1 -node <standbyhost1>
[oracle@<standbyhost1>]$ srvctl add instance -database <standby unique name> -instance
<standby unique name>2 -node <standbyhost2>

***Repeat 'add instance' as needed for each instance of the standby database
```

9. ***Place the Standby Password File***

First create the database directory in the DATA disk group then place the password file copied from the primary database to /tmp

```
[oracle@<standbyhost1>]$ export ORACLE_HOME=/u01/app/12.2.0.1/grid <- Grid Home
[oracle@<standbyhost1>]$ export ORACLE_SID=+ASM1
[oracle@<standbyhost1>]$ export PATH=$ORACLE_HOME/bin:$PATH

[oracle@<standbyhost1>]$ asmcmd mkdir +DATAC1/<standby unique name>
[oracle@<standbyhost1>]$ asmcmd mkdir +DATAC1/<standby unique name>/PASSWORD

[oracle@<standbyhost1>]$ asmcmd --privilege sysdba pwcopy -f -dbuniquename <standby
db_unique_name> /tmp/orapw<standby unique name> +DATAC1/<standby unique
name>/PASSWORD/orapw<standby unique name>
*** This command will update clusterware for the database's pwfile location
```

10. ***Modify PFILE Parameters***

Change the pfile copied to the standby in step 5 /tmp/standby.pfile update the instance specific RAC parameters, db_unique_name. For example:

NOTE: This list is not exhaustive. There are many parameters that may need to be changed due to a change in db_unique_name or disk group names or db_domain. Review each parameter in the pfile and change as appropriate.

Parameters to be modified on the Standby as compared to the	
Primary - Only CONVERT parameters may change, the rest are for reference	**Standby -**
*.cluster_database=TRUE	*.cluster_database=TRUE
<primary unique name>2.instance_number=2	<standby unique name>2.inst
<primary unique name>1.instance_number=1	<standby unique name>1.inst
<primary unique name>2.thread=2	<standby unique name>2.thre
<primary unique name>1.thread=1	<standby unique name>1.thre
<primary unique name>2.undo_tablespace='UNDOTBS2'	<standby unique name>2.und
<primary unique name>1.undo_tablespace='UNDOTBS1'	<standby unique name>1.und
....	....
....	....

*.db_unique_name=<primary unique name> *.audit_file_dest=/u01/app/oracle/admin/<primary unique name>/adump *.log_archive_dest_1='LOCATION=USE_DB_RECOVERY_FILE_DEST VALID_FOR=(ALL_LOGFILES,ALL_ROLES) MAX_FAILURE=1 REOPEN=5 DB_UNIQUE_NAME=<primary unique name> ALTERNATE=LOG_ARCHIVE_DEST_10' *.log_archive_dest_10='LOCATION=+DATAC1 VALID_FOR=(ALL_LOGFILES,ALL_ROLES) DB_UNIQUE_NAME=<primary unique name> ALTERNATE=LOG_ARCHIVE_DEST_1' *.log_archive_dest_state_10='ALTERNATE' *.control_files='+DATAC1/<primary unique name>/CONTROLFILE/control.ctl' # *CONVERT parameters are not dynamic and require a restart of the database. *.LOG_FILE_NAME_CONVERT='+DATAC1/<standby unique name>','+DATAC1/<primary unique name>' *.DB_FILE_NAME_CONVERT='+DATAC1/<standby unique name>','+DATAC1/<primary unique name>'	*.db_unique_name=<standby *.audit_file_dest=/u01/app/ora *.log_archive_dest_1='LOCATI (ALL_LOGFILES,ALL_ROLES) M <standby unique name> ALTE *.log_archive_dest_10='LOCAT (ALL_LOGFILES,ALL_ROLES) D ALTERNATE=LOG_ARCHIVE_D *.log_archive_dest_state_10=' *.control_files='+DATAC1' *.LOG_FILE_NAME_CONVERT= name>','+DATAC1/<standby u *.DB_FILE_NAME_CONVERT=' name>','+DATAC1/<standby u **# For 12.2 and higher remo** **will be picked up by cluster**

NOTE: The database parameter db_name must be the same between primary and all standby database.

NOTE: The CONVERT parameters, log_file_name_convert and db_file_name_convert are not required for file name translation when Oracle Managed Files is used and the standby is on a different cluster than the primary. Setting LOG_FILE_NAME_CONVERT to some value enables online redo log pre-clearing which improves role transition performance.

NOTE: If disk group names are different between the primary and standby, change all disk group names accordingly.

11. ***Restore the Standby Controlfile from the Primary***

```
[oracle@<standbyhost1>]$ rman target /
RMAN> startup nomount pfile='/tmp/<pfile name>'
RMAN> restore standby controlfile from service '<primary unique nameprimary database service
name>'; <- the service name the TNS alias for the primary database

The 'output file name' from the restore will show the OMF file name for the controlfile of
the standby. This will be added to the pfile before creating an spfile.

For example:
Starting restore at <DATE>
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=<sid> instance=<instance> device type=DISK

channel ORA_DISK_1: starting datafile backup set restore
channel ORA_DISK_1: using network backup set from service <service>
channel ORA_DISK_1: restoring control file
channel ORA_DISK_1: restore complete, elapsed time: 00:00:10
output file name=<OMF controlfile name>  <-- THIS FILE NAME WILL BE ADDED TO THE PFILE
Finished restore at <DATE>

RMAN> alter database mount;
```

12. ***Update the controlfile name in the standby pfile***

Edit the pfile and change the controlfile name from the diskgroup name to the 'output file name' from the previous step.

```
control_files='<OMF controlfile name>'
```

13. ***Create the spfile***

From the edited pfile, create the spfile for the standby database.

```
[oracle@<standbyhost1>]$ sqlplus "/ as sysdba"

SQL> create spfile='+DATAC1/<standby unique name>/spfile<standby unique name>.ora' from
pfile='/tmp/standby.pfile';

File created.

*** This will update clusterware with the spfile location

SQL> shutdown immediate
```

Then mount the database (using the spfile and new standby controlfile):

```
[oracle@<standbyhost1>]$ srvctl start database -d <standby db_unique_name> -o mount
```

14. ***Configure Encryption (if needed)***

The standby database can be encrypted during instantiation when the primary is not encrypted. In this scenario, the TDE wallet is created on the primary database, copied to the standby and the restore command with the AS ENCRYPTED clause will encrypt datafiles as they are restored.

This is useful for Hybrid Data Guard configurations where the primary is on premises and the standby is in the cloud. Another use case is a fully encrypted configuration by encrypting the standby with RESTORE AS ENCRYPTED, executing a switchover to make the encrypted database the primary database, then using offline encryption to encrypt the original primary as a standby.

Step 1: Set the default encryption algorithm for the database

OFFLINE encryption uses the database default encryption algorithm, an algorithm cannot be passed with the encrypt command. By default for versions through 23ai, the default encryption algorithm is AES128. If a different algorithm is desired, set "_tablespace_encryption_default_algorrithm" accordingly on both the primary and standby databases. This must be done prior to the first SET KEY command to set the master encryption key.

```
ALTER SYSTEM SET "_TABLESPACE_ENCRYPTION_DEFAULT_ALGORITHM"=AES256 SCOPE = BOTH SID = '*';

For 21c the parameter name changes to TABLESPACE_ENCRYPTION_DEFAULT_ALGORITHM

ALTER SYSTEM SET "TABLESPACE_ENCRYPTION_DEFAULT_ALGORITHM"=AES256 SCOPE = BOTH SID = '*'; --
for 21c and later
```

For a full list of supported algorithms, refer to Supported Encryption and Integrity Algorithms for the appropriate database version.

WARNING: For reasons of compatibility between versions, if the master key was created/set in database version 11.1, after upgrading to 11.2 or higher the master key should be re-keyed.

Step 2: Configure the Keystore on the Primary Database

The keystore must be created in a read-write database and is therefore configured on the primary database. The standby database then must have access to that keystore either through OKV or in the case of file based keystores, the files must be copied to the standby keystore location.

Follow the steps in Configuring Transparent Data Encryption for the desired type of keystore.

Return to this document when you have reached the step for 'Encrypt Your Data'

Step 3: Configure Standby Access to Keystore

The standby database needs access to the same keystore as was created on the primary. In the case of file-based keystores, the files must be copied to the standby's <WALLET_ROOT>/tde location. Refer to the documentation for configuring standby access for other methods.

15. ***Configure degree of parallelism***

To take advantage of parallelism during the restore, determine the number of cpu's on your server by executing the following:

```
[oracle@<standbyhost1>]$ grep -c ^processor /proc/cpuinfo
```

Make the following RMAN configuration changes at the standby. Set the parallelism to match results of the network evaluation to achieve the best performance.

```
[oracle@<standbyhost1>]$ rman target /
RMAN> CONFIGURE DEFAULT DEVICE TYPE TO DISK;
RMAN> CONFIGURE DEVICE TYPE DISK PARALLELISM 8;
```

Large Database Optimizations

The two topics below, instance parallelism and gap resolution with recover from service, are optimizations for large databases. While this note without these optimizations is relevant for most cases without these optimizations, these optimizations can significantly shorten the time required to instantiate very large databases.

Instance Parallelism - Using multiple channels with the PARALLELISM setting is helpful in utilizing the available resources in a single node. This is generally enough for small to medium sized databases measured in the hundreds of gigabytes or less. For larger databases, spreading the RMAN work across multiple nodes/instances of a RAC cluster can utilize additional resources and bandwidth across multiple nodes of a cluster. This process is described further in the next step.

Gap Resolution with RECOVER DATABASE FROM SERVICE - While the restore database from service is running, the gap in redo to make the database consistent is growing. The first data file is copied as of time while the last data file is copied as of a later time. Therefore, when the restore is complete, the database is not consistent and requires redo from the primary database to catch up. In most cases, simply enabling redo transport will resolve the gap in a reasonable amount of time. However, there are cases for which transport/apply would be either a complicated or lengthy process. For example:

1. The restore took many hours
2. There was a lot of redo generated while the restore was taking place
3. Archived logs were backed up and removed from the primary recovery area

For these situations, it is often more efficient to utilize RECOVER DATABASE FROM SERVICE to 'roll forward' the standby database. The recovery process is optimized to only copy the blocks that have changed.

The steps to execute recover database from service is described at the appropriate point in the process below.

For more details see the RMAN documentation

16. ***Restore the Standby Database from the primary database service***

Restoring the datafiles from the primary to standby for instantiation is initiated on the standby cluster. Maximizing the use of the channels configured for parallelization by the previous step can be improved in some cases by using the RMAN SECTION SIZE clause.

Section Size

Parallelization across channels is most efficient when the datafiles are all equally sized. Since each file by default is copied by only one channel, if one or two files are significantly larger than the rest, those files will take longer while other files have finished leaving some channels idle. When a small subset of files is larger than the rest, allowing RMAN to copy those large files in sections can utilize all channels to maximum efficiency. The section size can be set with the RMAN SECTION SIZE clause. RMAN testing has shown SECTION SIZE=64G to provide the best efficiency for files less than 16TB. If the data file is smaller than the section size chosen, it will not be broken into sections during the restore.

On the primary, query the largest datafile size to determine the section size to be used for the recover command.

```
SQL> select max(bytes)/1073741824 GB from v$datafile;
```

If the largest file is:
<15TB use section size of 64G
>15TB and <30TB used section size of 128G
>30TB and <60TB used section size of 256G
>60TB use section size of 512G

For more information please refer to documentation RMAN 'restore from service'. Also see refer to the best practice for Cloud.

RESTORE DATABASE FROM SERVICE

In order to initiate the copy of files, connect to the standby database and issue the restore command below using the descriptor in the tnsnames.ora for the primary database. In this example, that is the primary db_unique_name.

```
[oracle@<standbyhost1>]$ rman target /

RMAN> restore database from service <primary unique name> section size <section size> [AS ENCRYPTED];

RMAN> backup spfile;

RMAN> switch database to copy;   <- This may result in a no-op (nothing executes)
```

NOTE: If the restore database from service fails...

Re-running the restore database from service command will recopy all files whether they were completed, partially completed or not started. In the event of a failed attempt this could result in multiple copies of files and for large databases can result in significant time lost recopying files unnecessarily. If the restore database from service fails, remove the partial files and replace the restore database from service command with a restore datafile from service listing all files not yet completed.

From the mounted standby (as sys) list the partial files with the following SQL and remove them from the file system:

```
select HXFNM from x$kcvfh where FHTYP=11;
```

Then replace the restore database from service with the restore datafile from service generated with the SQL below:

```
select 'restore datafile '||listagg(FILE#,', ') within group (order by file#)||' from
service <primary unique name> section size 64G;' from v$datafile_header where ERROR='FILE
NOT FOUND';
```

The listagg function has size limits as does the RMAN command line. On a database with many files, if either is exceeded resulting in an error the list of files may need to be broken up into multiple commands.

This method should be used when using instance parallelism as well.

ENCRYPTION NOTE: For cases where the primary database is not encrypted and the standby is encrypted, the files should be restored using the AS ENCRYPTED clause. This clause is not valid on RESTORE DATAFILE so the tablespace(s) of the missing datafile(s) must be restored using RESTORE TABLESPACE instead.

For example: RESTORE TABLESPACE <tablespace name> FROM SERVICE <tns alias for primary database> SECTION SIZE <section size> AS ENCRYPTED;

Large Database Optimization - Instance Parallelism

For larger databases, instead of the previous code block the following can be used.

In order to parallelize the restore across all instances and utilize the bandwidth of all nodes or a cluster, use the connect clause in RMAN. In this method, the parallelization is created through allocating channels in the run block rather than the setting defined in the previous step.

If you've followed this note at this point all instances are not mounted. Stop the database and startup mount all instance, then execute the run block with the proper substitutions.

Allocate the number of channels to match results of the network evaluation to achieve the best performance.

For 4-node and 8-node clusters allocate additional channels connecting to those instances.

```
[oracle@<standbyhost1>]$ srvctl stop database -db <standby unique name>
[oracle@<standbyhost1>]$ srvctl start database -db <standby unique name> -startoption mount

[oracle@<standbyhost1>]$ rman target sys/<password> <- It is necessary to connect with the password
RMAN > run {
allocate channel c1 type disk connect '/@<standby instance 1 SID_NAME>';
allocate channel c2 type disk connect '/@<standby instance 1 SID_NAME>';
allocate channel c3 type disk connect '/@<standby instance 1 SID_NAME>';
allocate channel c4 type disk connect '/@<standby instance 1 SID_NAME>';
allocate channel c5 type disk connect '/@<standby instance 2 SID_NAME>';
allocate channel c6 type disk connect '/@<standby instance 2 SID_NAME>';
allocate channel c7 type disk connect '/@<standby instance 2 SID_NAME>';
allocate channel c8 type disk connect '/@<standby instance 2 SID_NAME>';
restore database from service '<primary unique name>' section size 64G [AS ENCRYPTED];
}

RMAN> backup spfile;

RMAN> switch database to copy;  <- This is not always necessary so may result in a no-op (nothing executes)
```

NOTE: For larger clusters or a higher degree of parallelism, allocate additional channels accordingly.

Large Database Optimization - Gap Resolution with RECOVER DATABASE FROM SERVICE

For large databases which took a long time to copy, generated a lot of redo during the copy, or do not have the archived logs available at the primary since the copy started, use this RECOVER DATABASE FROM SERVICE option before enabling Data Guard Broker in the next step. For databases which do not meet that description, skip to the next step.

```
[oracle@<standbyhost1>]$ srvctl stop database -d <standby unique name>

[oracle@<standbyhost1>]$ rman target /

RMAN > startup nomount;
RMAN > restore standby controlfile from service <primary unique name>;

RMAN > alter database mount;

RMAN > catalog start with '<DATA DISK GROUP>' noprompt; <-- This step catalogs all files into the copied controlfile.

RMAN > catalog start with '<RECO DISK GROUP>' noprompt; <-- Some files cannot be cataloged and will generate an error.

RMAN > switch database to copy;

RMAN > shutdown immediate;

RMAN > exit;

[oracle@<standbyhost1>]$ srvctl start database -d <standby unique name> -o mount
```

This process uses the instance parallelization method, normal parallelization where all channels run on one instance can also be used.

Allocate the number of channels to match results of the network evaluation to achieve the best performance.

For 4-node and 8-node clusters allocate additional channels connecting to those instances.

```
-- All database instances should be mounted

[oracle@<standbyhost1>]$ rman target sya/<password>  <- It is necessary to connect with the password

RMAN > run {
allocate channel c1 type disk connect '/@<standby instance 1 SID_NAME>';
allocate channel c2 type disk connect '/@<standby instance 1 SID_NAME>';
allocate channel c3 type disk connect '/@<standby instance 1 SID_NAME>';
allocate channel c4 type disk connect '/@<standby instance 1 SID_NAME>';
allocate channel c5 type disk connect '/@<standby instance 2 SID_NAME>';
allocate channel c6 type disk connect '/@<standby instance 2 SID_NAME>';
allocate channel c7 type disk connect '/@<standby instance 2 SID_NAME>';
allocate channel c8 type disk connect '/@<standby instance 2 SID_NAME>';
recover database from service '<primary unique name>' section size 64G;
}

RMAN> switch database to copy;  <- This is not always necessary so may result in a no-op (nothing executes)

RMAN > exit;
```

NOTE: This process can be used any time a standby has an unresolvable or large gap as a means of catching up to the primary. See MOS Note 2850185.1 for a full description of the process on an established standby database.

17. ***Clear Logfiles***

Clear all Online Redo Logs and Standby Redo Logs

```
[oracle@<standbyhost1>]$ sqlplus / as sysdba

begin
for log_cur in ( select group# group_no from v$log )
loop
execute immediate 'alter database clear logfile group '||log_cur.group_no;
end loop;
end;
/

begin
for log_cur in ( select group# group_no from v$standby_log )
loop
execute immediate 'alter database clear logfile group '||log_cur.group_no;
end loop;
end;
/
```

18. ***Set the parameters and create the Data Guard Broker configuration.***

Modify the script below to your environment and save as PostCR.sql

NOTE: These commands can also be executed individually from sqlplus as sys

```
connect / as sysdba

alter system set dg_broker_config_file1='+DATAC1/<standby unique name>/dr1.dat' scope=both
sid='*';

alter system set dg_broker_config_file2='+RECOC1/<standby unique name>/dr2.dat' scope=both
sid='*';

alter system set dg_broker_start=true scope=both sid='*';

alter system register;

connect sys/<password>@<primary unique name> as sysdba

alter system set dg_broker_config_file1='+DATAC1/<primary unique name>/dr1.dat' scope=both
sid='*';

alter system set dg_broker_config_file2='+RECOC1/<primary unique name>/dr2.dat' scope=both
sid='*';

alter system set dg_broker_start=TRUE;

host sleep 30

host dgmgrl sys/<password>@<primary unique name> "CREATE CONFIGURATION dgconfig AS PRIMARY
DATABASE IS <primary unique name> CONNECT IDENTIFIER IS <primary unique name>";

host sleep 30

host dgmgrl sys/<password>@<primary unique name> "ADD DATABASE <standby unique name> AS
CONNECT IDENTIFIER IS <standby unique name>" ;

host dgmgrl sys/<password>@<primary unique name> "ENABLE CONFIGURATION"

exit
```

Execute the script PosrtCR.sql on the standby database. Set your environment to standby database.

```
[oracle@<standbyhost1>]$ export ORACLE_HOME=/u01/app/oracle/product/12.2.0.1/dbhome_1
[oracle@<standbyhost1>]$ export ORACLE_SID=<standby unique name>1
[oracle@<standbyhost1>]$ export PATH=$ORACLE_HOME/bin:$PATH
[oracle@<standbyhost1>]$ sqlplus / as sysdba

SQL> @PostCR.sql
```

19. ***Stop and Start the standby database***

```
[oracle@<standbyhost1>]$ srvctl stop database -db <standby unique name> -o immediate
[oracle@<standbyhost1>]$ srvctl start database -db <standby unique name>
```

20. ***Validate broker configuration***

```
[oracle@<standbyhost1>]$dgmgrl sys/<password>

DGMGRL> show configuration

Configuration - dgconfig

Protection Mode: MaxPerformance
Members:
<primary unique name> - Primary database
<standby unique name> - Physical standby database

Fast-Start Failover: DISABLED
```

```
Configuration Status:
SUCCESS (status updated 58 seconds ago)
```

Implement MAA Best Practice Recommendations

The following settings are recommended per MAA best practices and should be set on the primary and standby databases:

Data Protection Parameters

The parameters below help actively protect against data corruption.

DB_BLOCK_CHECKING=MEDIUM or higher

NOTE: DB_BLOCK_CHECKING can have performance implications on a primary database. Any changes to this setting should be thoroughly tested before implementing.

DB_BLOCK_CHECKSUM=TYPICAL or higher

DB_LOST_WRITE_PROTECT=TYPICAL

Enable Flashback Database

Flashback database is required to reinstate a failed primary after a failover.

It is an MAA recommendation but there are some performance implications and should be tested to determine if the impact to the performance of the application is acceptable.

NOTE: Without flashback enabled the primary database must be fully re-instantiated after a failover using another restore from service. Switchover does not require flashback database.

Primary:

```
sqlplus / as sysdba

SQL> alter database flashback on;
```

To enable flashback database on the standby the redo apply process must first be stopped. Once flashback has been enabled redo apply can be restarted:

```
DGMGRL> CONNECT sys/<password>@<standby unique name>
DGMGRL> EDIT DATABASE <standby unique name> SET STATE=APPLY-OFF;
DGMGRL> SQL "ALTER SYSTEM SET DB_FLASHBACK_RETENTION_TARGET=1440 SCOPE=BOTH SID='*'";
DGMGRL> SQL "ALTER DATABASE FLASHBACK ON";
DGMGRL> EDIT DATABASE <standby unique name> SET STATE=APPLY-ON;
```

Didn't find what you are looking for?