# Creating a Standby using RMAN Duplicate (RAC or Non-RAC) (Doc ID 1617946.1)

**In this Document**

Goal
Solution
R

## APPLIES TO:

Oracle Database - Ent Edition - Version 11.2.0.0 to 12.1.0.2 [Release 11.2 to 12.1]
Information in this document applies to any platform.

## GOAL

### 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

The purpose of this document is to provide a step-by-step guide for creating a standby database in an 11.2 or 12c environment integrating the MAA Data Guard configuration best practices. This paper assumes that the following conditions exist:

1. A Primary RAC or single instance database utilizing ASM for data file storage
2. The Primary database is in archive log mode
3. All Standby target hosts have existing Oracle software installation
4. The Standby target database storage will utilize ASM

- It is recommended that you consult the Data Guard Concepts and Administration guide as well as the HA Best Practice guide for more information and full MAA Best Practices.

### Supported Versions

This document applies to both Oracle Server versions 11.2.0.x and 12.1.0.x or higher. There are some important differences in how the DUPLICATE FOR STANDBY FROM ACTIVE DATABASE functions between the two releases which are noted below:

11.2:

- DUPLICATE FROM ACTIVE DATABASE uses datafile image copies and does not support section size, compression, or encryption.

12c:

- DUPLICATE FROM ACTIVE DATABASE supports backup sets.
- SECTION SIZE support is available. If section size is used, then use multiple auxiliary channels for parallelism.
- Compression is supported but do not 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 in elapsed time) can be significant.
- Encryption is supported.

All of the examples illustrated in this document use the following naming:

| | **Primary** | **Standby** |
|---|---|---|
| **Hosts** | exa503, exa504 | exa505, exa506 |
| **Database Unique Name** | chicago | boston |
| **Instance names** | chicago1, chicago2 | boston1, boston2 |

## SOLUTION

**Prerequisite:**

Performing an active database RMAN duplicate means that multiple channels can be spread across primary nodes. By doing this we can utilize more network interfaces for the pull/push operations.

Parallelize backups across all primary database nodes leveraging disk and network bandwidth as available.

MAA best practice recommend using 8 channels minimum or 2 channels per node.

Use automatic load balancing to distribute RMAN channels among the allocated nodes. With auto load balancing the channel distribution is approximate point in time. Analyze the impact on existing production databases and restrict the usage if necessary.

Make the following RMAN configuration changes at the Primary. Our example uses 16 preconfigured channels for RMAN to use during the standby creation step. These 16 channels will suffice for 2-4 node RAC clusters but will need to be increased if your Production RAC cluster is larger than 4 nodes.

```
RMAN> CONFIGURE DEFAULT DEVICE TYPE TO DISK;

RMAN> CONFIGURE DEVICE TYPE DISK PARALLELISM 16;
```

You will use Oracle Net service load balancing to distribute these RMAN channels evenly among the allocated instances.

1) Create a service to run RMAN on allocated instances in the cluster:

```
srvctl add service -db chicago -service srv_rman -preferred chicago1,chicago2

srvctl start service -db chicago -service srv_rman
```

Note: If you need to restrict the RMAN process to specific instances only specify those instances in the service definition

2) When running RMAN, use the service name in the connect string for the "target" parameter:

```
rman target sys/<password>@<prim_scan>/srv_rman
```

**Steps to Duplicate the Primary Database**

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

1. Create standby redo logs on the primary database that are the same size of the online redo logs. This will ensure that standby redo log files are automatically created at the standby during the RMAN duplication process and that standby redo log files are available on the current Primary after a role transition occurs in the future and the Primary becomes a standby.

Oracle recommends having the same number of standby redo log files as there are online redo log files plus one additional standby redo log for each thread. Our primary database has 6 online redo log files, 3 per thread. We therefore need 4 Standby Redo Log files per thread (Primary 3 plus 1) for a total of 8 Standby Redo Log files. For example:

```
SQL> alter database add standby logfile thread 1
group 7 size 500M,
group 8 size 500M,
group 9 size 500M,
group 10 size 500M;

SQL> alter database add standby logfile thread 2
group 11 size 500M,
group 12 size 500M,
group 13 size 500M,
group 14 size 500M;
```

2. Database force logging is recommended as an MAA Data Guard best practice. To enable force logging, use the following command on the primary:

```
SQL> alter database force logging;
```

3. For the RMAN duplication process it is necessary to setup a temporary listener which will later be removed.

In the standby database home, create and start a listener that offers a static SID entry for the standby database with the ORACLE_SID for the standby (boston1) and ORACLE_HOME for the standby.

```
LISTENER_duplicate =
(DESCRIPTION_LIST =
(DESCRIPTION =
(ADDRESS = (PROTOCOL = TCP)
(HOST = exa505)
(PORT = 1525)(IP = FIRST))))

SID_LIST_LISTENER_duplicate =
(SID_LIST =
(SID_DESC =
(SID_NAME = boston1)
(ORACLE_HOME = /u01/app/oracle/product/11.2.0/db_1)))
```

4. In the database home on the primary node, create an Oracle Net alias to connect to the listener created in the above step.

```
dup =
(DESCRIPTION =
(ADDRESS =
(PROTOCOL = TCP)
(HOST = exa505)
(PORT = 1525))
(CONNECT_DATA =
(SERVER = DEDICATED)
(SID = boston1)))
```

5. For active database duplication, you must create a password file for the auxiliary instance and establish Oracle Net connectivity. This is a temporary password file as it will be overwritten during the duplicate operation.

Create a password file on the standby host in the $ORACLE_HOME/dbs directory with the same SYS (or SYSDG if you use that username in 12c) as the Primary database. For example:

```
orapwd file=orapwboston1 password=<primary database sys password>
```

6. On the standby host in the ORACLE_HOME/dbs directory create a pfile (initboston1.ora) with the following parameters. It is recommended to set the sga_t . For example:

```
db_name=chicago
db_unique_name=boston
sga_target=5g
```

7. On all standby hosts create the audit directory for the boston database:

```
mkdir -p /u01/app/oracle/admin/boston/adump
```

8. On all primary hosts create an Oracle Net alias to reach the boston database on the standby nodes. Make sure that all hosts have both a chicago and boston Oracle Net alias and that all the aliases reference the scan listener and not the node vip. Also, if local_listener is set to an alias on the primary create a corresponding entry on the standby side that points to the local listener on that system. For example:

| In all tnsnames.ora files on the primary nodes | In all tnsnames.ora files on the standby nodes |
|---|---|
| chicago =<br>(DESCRIPTION =<br>(ADDRESS_LIS<br>ESS=(PROTOCOL= TCP)(HOST=prmy-scan)<br>(PORT = 1521))<br>)<br>(CONNECT_DATA =<br>(SERVER = DEDICATED)<br>(SERVICE_NAME = chicago)<br>)<br>)<br><br>boston =<br>(DESCRIPTION =<br>(ADDRESS_LIS<br>T=stby-scan) | chicago =<br>(DESCRIPTION =<br>(ADDRESS_LIS<br>TOCOL= TCP)(HOST=prmy-scan)<br>(PORT = 1521))<br>)<br>(CONNECT_DATA =<br>(SERVER = DEDICATED)<br>(SERVICE_NAME = chicago)<br>)<br>)<br><br>boston =<br>(DESCRIPTION =<br>(ADDRESS_LIST =<br>(ADDRESS=(PROTOCOL = TCP)(HOST=stby-scan) |

| | |
|---|---|
| (PORT = 1521))<br>)<br>(CONNECT_DATA =<br>(SERVER = DEDICATED)<br>(SERVICE_NAME = boston)<br>)<br>)<br><br>chicago_local_listener =<br>(DESCRIPTION =<br>(ADDRESS_LIS<br>TOCOL = TCP)(HOST=prmy-vip)<br>(PORT = 1521))<br>)<br>) | (PORT = 1521))<br>)<br>(CONNECT_DATA =<br>(SERVER = DEDICATED)<br>(SERVICE_NAME = boston)<br>)<br>)<br><br>boston_local_listener =<br>(DESCRIP ION =<br>(ADDRESS_LIS<br>T=stby-vip)<br>(PORT = 1521))<br>)<br>) |

9. On the standby host set the SID to the standby SID (boston1) and startup nomount the standby/auxiliary instance.

```
$ export ORACLE_SID=boston1
SQL> startup nomount pfile='initboston1.ora'
```

10. If the cluster_interconnects parameter is set on the primary it will be necessary to temporarily unset it in the SPFILE only. It will be returned to the original v lues in Step 13.

> **Note:** Do NOT restart any primary instance. The reset is done to temporarily remove the cluster_interconnects parameter from the spfile so that when the SPFILE is copied to the standby system it will avoid any issues if the standby database uses the same instance names as the primary database.

The values of cluster_interconnects for each instance should be noted before resetting the SPFILE as they will be replaced after standby has been created. For example, execute the following commands and note down the values for each instance in the Primary cluster.

```
SQL> select p.inst_id,instance_name, name,value from gv$parameter p, gv$instance i where
p.inst_id=i.inst_id and p.name='cluster_interconnects';

INST_ID      INSTANCE_NAME    NAME                                          VALUE
---------- ---------------- ---------------------------------------------- ------------------
-------
1            chicago1            cluster_interconnects
192.168.10.149
2            chicago2            cluster_interconnects
192.168.10.150
```

Now reset the cluster_interconnects parameter temporarily in the SPFILE only:

```
SQL> alter system reset cluster_interconnects scope=spfile sid='chicago1';
SQL> alter system reset cluster_interconnects scope=spfile sid='chicago2';
```

11. On the primary host run an RMAN script that duplicates the primary database using the "duplicate target database for standby from active database" command. Note that the contents of the duplicate command will v . The following example covers the majority of use cases. For any case not covered by the examples please consult chapter 24 of the Oracle® Database Backup and Recovery User's Guide for

complete information.

The first example illustrates how to duplicate between two systems where the ASM diskgroup names are the same:

```
rman <<EOF
connect target sys/<password>@<prim_scan>/srv_rman;
connect auxiliary sys/<password>@dup;
run {
duplicate target database for standby from active database
spfile
parameter_value_convert 'chicago','boston'
set db_unique_name='boston'
SET CLUSTER_DATABASE='FALSE'
set control_files='+DATA/boston/standby.ctl'
set local_listener='boston_local_listener'
set remote_listener='stby-scan:1521'
set audit_file_dest='/u01/app/oracle/admin/boston/adump';
}
EOF
```

The following example illustrates how to duplicate between two systems where the source is on a file system and the target is using an ASM diskgroup. The same process will work where both databases use ASM but they have different Disk Group names.

In our example we use the file system names for the Primary but if this was ASM to ASM with different disk group names you would substitute the Primary Disk group name.

```
rman <<EOF
connect target sys/<password>@<prim_scan>/srv_rman;
connect auxiliary sys/<password>@dup;
run {
duplicate target database for standby from active database
spfile
parameter_value_convert 'chicago','boston'
set db_file_name_convert '/u01/data/','+DATA'
set db_unique_name='boston'
SET CLUSTER_DATABASE='FALSE'
set db_create_online_log_dest_1='+DATA'
set db_create_file_dest='+DATA'
set db_recovery_file_dest='+RECO'
set log_file_name_convert '/u01/data/','+DATA','/u01/reco/','+RECO'
set control_files='+DATA/boston/standby.ctl'
set local_listener='boston_local_listener'
set remote_listener='stby-scan:1521'
set audit_file_dest='/u01/app/oracle/admin/boston/adump';
}
EOF
```

12. Reset CLUSTER_INTERCONNECTS to original v lue from Step 11 in the SPFILE with the following commands. Do NOT restart any primary instance.

```
SQL> alter system set cluster_interconnects='192.168.10.149' scope=spfile sid='chicago1';
SQL> alter system set cluster_interconnects='192.168.10.150' scope=spfile sid='chicago2';
```

13. Stop and remove the listener created in step 3. Also remove the TNS entry created in Step 4.

14. Copy the password file to the respective location.

If the standby is version 11.2 then no action is required.

If the standby database is at Version 12.1 or higher copy the password file to ASM. For Ex le:

```
$asmcmd -p
ASMCMD [+] > cd +DATA
AMSCMD [+DATA] >mkdir BOSTON/PASSWORD
ASMCMD [+DATA] >pwcopy /u01/app/oracle/product/12.1.0.2/dbhome_1/dbs/orapwboston1
+DATA/BOSTON/PASSWORD/pwboston
ASMCMD [+DATA] > exit

Remove the original password file.

$rm /u01/app/oracle/product/12.1.0.2/dbhome_1/dbs/orapwboston1
```

15. Create the standby spfile in ASM.

Create a spfile in +DATA for the standby database:

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

SQL> create spfile='+DATA/boston/spfileboston.ora' from '/tmp/p.ora'

$rm /u01/app/oracle/product/12.1.0.2/dbhome_1/dbs/spfileboston1.ora
```

16. On standby host create an initboston1.ora file that points to the spfile created in the above step.

```
$cat initboston1.ora
spfile='+DATA/boston/spfileboston.ora'
```

17. Restart the standby instance. Then register the database with CRS:

```
SQL>shutdown immediate
SQL> startup mount

11.2:
srvctl add database -d boston -o /u01/app/oracle/product/11.2.0/db_1 -x exa505 -i boston1
srvctl modify database -d boston -r physical_standby -p '+DATA/boston/spfileboston.ora'

12.1 :
srvctl add database -db boston -oraclehome /u01/app/oracle/product/12.1.0.2/dbhome_1 -node
exa505 -instance boston1
srvctl modify database -db boston -role physical_standby -spfile
'+DATA/boston/spfileboston.ora' -pwfile '+DATA/BOSTON/PASSWORD/pwboston'
```

NOTE:If your standby database is single instance, the standby database configuration is complete and you should proceed to the section: Creating a Data Guard Broker configuration.
If y lete Standby RAC Configuration.

**Complete Standby RAC Configuration**

After completing the steps in the above section the standby database has been created on the standby host. When the primary is RAC configuration then a few configuration items need to be completed to finish the RAC configuration on the standby. These remaining configuration steps are covered below.

1. Create a temporary pfile from spfile on the standby.

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

2. Modify the parameters on the Standby to update the instance specific RAC parameters. For example:

| Primary | Standby |
|---|---|
| *.cluster_database=TRUE<br>chicago2.instance_number=2<br>chicago1.instance_number=1<br>chicago2.thread=2<br>chicago1.thread=1<br>chicago2.undo_t lespace='UNDOTBS2'<br>chicago1.undo_t lespace='UNDOTBS1'<br>…….<br>……. | *.cluster_database=TRUE<br>boston2.instance_number=2<br>boston1.instance_number=1<br>boston2.thread=2<br>boston1.thread=1<br>boston2.undo_t lespace='UNDOTBS2'<br>boston1.undo_t lespace='UNDOTBS1'<br>…….<br>……. |

3. Create a spfile in +DATA for the standby database:

```
SQL> create spfile='+DATA/boston/spfileboston.ora' from pfile='/tmp/p.ora';
```

4. Copy the password file to the respective location.

If the standby database is version 11.2, you must copy the password file to the $ORACLE_HOME/dbs directory on all the other standby hosts and name it per the standby SID on each host. For example on node2 you would name it $ORACLE_HOME/dbs/orapwboston2.

If the standby database is version 12.1 then no action is needed. The password was copied to ASM in Step 14, which is visible to all nodes across the cluster.

5. On all standby hosts create an initboston<SID Number>.ora file that points to the spfile created in the above step. For example:

```
$cat initboston1.ora
spfile='+DATA/boston/spfileboston.ora'
```

6. Restart all standby instances. R .

```
SQL>shutdown immediate


11.2
srvctl add database -d boston –o /u01/app/oracle/product/11.2.0/db_1
srvctl add instance -d boston -i boston1 -n exa505
srvctl add instance -d boston -i boston2 -n exa506
srvctl modify database –d boston –r physical_standby -p '+DATA/boston/spfileboston.ora'

12.1 :
srvctl add database -db boston -oraclehome /u01/app/oracle/product/12.1.0.2/dbhome_1
srvctl add instance -db boston -instance boston1 -node exa505
srvctl add instance -db boston -instance boston2 -node exa506
srvctl modify database -db boston -role physical_standby -spfile
'+DATA/boston/spfileboston.ora' -pwfile '+DATA/BOSTON/PASSWORD/pwboston'
```

```
Restart standby database on all standby hosts:
srvctl start database -d boston -o mount
```

**Creating a Data Guard Broker configuration:**

The following section describes the basic steps on how to create a Data Guard broker configuration. For complete information on the Data Guard broker consult the Oracle® Data Guard Broker guide.

1. On both the primary and standby configure the Data Guard broker metadata files and enable the broker:

Primary:

```
SQL> alter system set dg_broker_config_file1='+DATA/chicago/dr1.dat' scope=both;
SQL> alter system set dg_broker_config_file2='+RECO/chicago/dr2.dat' scope=both;
SQL> alter system set dg_broker_start=true scope=both;
```

Standby

```
SQL> alter system set dg_broker_config_file1='+DATA/boston/dr1.dat' scope=both;
SQL> alter system set dg_broker_config_file2='+RECO/boston/dr2.dat' scope=both;
SQL> alter system set dg_broker_start=true scope=both;
```

2. Add static sid entries into the local node listener.ora located in the grid infrastructure home on all hosts in the configuration. Please refer to <Note 1387859.1> for instructions on how to complete this):

> Note: Static "_DGMGRL" entries are no longer needed as of Oracle Database 12.1.0.2 in Oracle Data Guard Broker configurations that are managed by Oracle Restart, RAC One Node or RAC as the Broker will use the clusterware to restart an instance.

For example:

```
LISTENER_SCAN2=(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=IPC)(KEY=LISTENER_SCAN2))))
LISTENER_SCAN3=(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=IPC)(KEY=LISTENER_SCAN3))))
LISTENER=(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=IPC)(KEY=LISTENER))))

LISTENER_SCAN1=(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=IPC)(KEY=LISTENER_SCAN1))))
ENABLE_GLOBAL_DYNAMIC_ENDPOINT_LISTENER_SCAN1=ON
ENABLE_GLOBAL_DYNAMIC_ENDPOINT_LISTENER=ON
SID_LIST_LISTENER =
(SID_LIST =
(SID_DESC =
(GLOBAL_DBNAME =boston_DGMGRL)
(SID_NAME =boston1)
(ORACLE_HOME = /u01/app/oracle/product/11.2.0/db_1)))
ENABLE_GLOBAL_DYNAMIC_ENDPOINT_LISTENER_SCAN3=ON
```

Note that each static entry references the SID for each node, boston1, boston2 or chicago1, chicago2 for an ADMIN managed RAC cluster. If the RAC is a Policy Managed RAC please refer to the note above for specific instructions on static entries in 11g Release 2.

3. Bounce or reload all the listeners where the above modification was made (primary and standby nodes):

```
srvctl stop listener
srvctl start listener
```

4. On a primary host connect with dgmgrl and create the configuration:

```
[oracle@exa503 /etc]$ dgmgrl sys/password

DGMGRL> create configuration 'dg_config' as  primary database is 'chicago' connect
identifier is chicago;

Configuration "dg_config" created with primary database "chicago"

DGMGRL> add database 'boston' as connect identifier is boston;

Database "boston" added

DGMGRL> enable configuration;

Enabled.
```

5. Verify that the configuration created successfully by using the show configuration command:

```
DGMGRL> show configuration;

Configuration - dg_config

Protection Mode: MaxPerformance
Databases:
chicago - Primary database
boston - Physical standby database

Fast-Start Failover: DISABLED

Configuration Status:
SUCCESS
```

6. Flashback database is required to reinstate a failed primary after a failover role transition. Optionally enable flashback on both the primary and standby:

Primary:

```
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:

```
SQL> recover managed standby database cancel;
SQL> alter database flashback on;
SQL> recover managed standby database disconnect using current logfile;
```

The above steps can also be accomplished using the broker with the following commands. (you must be connected to the standby database via DGMGRL):

```
DGMGRL> CONNECT sys/<password>@boston
DGMGRL> EDIT DATABASE boston SET STATE=APPLY-OFF;
DGMGRL> SQL "ALTER DATABASE FLASHBACK ON";
DGMGRL> EDIT DATABASE boston SET STATE=APPLY-ON;
```

**Application Service Considerations with Data Guard:**

In an Oracle Data Guard configuration, applications must use Services to connect to the databases as there is more

than one database with different SID names that could be the Primary database. And with Oracle Active Data Guard, any physical standby database could be made available to provide read only access to the data. To enable client connections, services need to be created for the workload that performs updates which will only be started on the database in the configuration that is currently in the 'Primary' role as well as services for read only reports and queries that should only be present on physical standbys that are open in Active Data Guard.

> Note: Refer to MOS Note 2123709.1 for full Client Failover configuration MAA Best Practices.

The following is an example of how to configure a Read/Write service (workload) and a Read/Only service (reports) on the Primary and Standby database.

If the primary / standby is configured with Oracle Restart, RAC One Node or RAC, the services are created using srvctl. (Note for JDBC application server side TAF attributes should be set to NONE). If you are creating services for a multitenant database then use service commands that include the –pdb option.

For example, here we add show adding two services to each database in the configuration. WORKLOAD for Read/Write connections and REPORTS for Read/Only connections. WORKLOAD will only run on the database that is running in the Primary role and REPORTS will only run on the database that is in the PHYSICAL STANDBY role.

Primary cluster:

```
srvctl add service -d chicago -s workload -r chicago1,chicago2 -l PRIMARY -q TRUE -e SESSION
-m BASIC -w 10 -z 150
srvctl add service -d chicago -s reports -r chicago1,chicago2 -l PHYSICAL_STANDBY -q TRUE -e
SESSION -m BASIC -w 10 -z 150
srvctl start service -d chicago -s reports
srvctl stop service -d chicago -s reports
```

> Note: We need to start and stop the REPORTS services on the Primary to create the service in the database so that it can be started on the Standby. If we do not first start it on the Primary database the service will not be in the database. Services metadata and a start of that service on the standby will fail because it needs to create the service, which it cannot do in a read only database.

Standby Cluster:

```
srvctl add service -d boston -s workload -r boston1,boston2 -l PRIMARY -q TRUE -e SESSION -m
BASIC -w 10 -z 150
srvctl add service -d boston -s reports -r boston1,boston2 -l PHYSICAL_STANDBY -q TRUE -e
SESSION -m BASIC -w 10 -z 150
srvctl start service -d chicago -s reports
```

2. Create Oracle Net aliases that reference the above services for the client applications to use.

```
WORKLOAD=
(DESCRIPTION =
(ADDRESS_LIST =
(ADDRESS=(PROTOCOL= TCP)(HOST=prmy-scan)(PORT = 1521)))
(CONNECT_DATA =
(SERVER = DEDICATED)
(SERVICE_NAME = workload)
)
```

```
)


REPORTS=
(DESCRIPTION =
(ADDRESS_LIST =
(ADDRESS=(PROTOCOL= TCP)(HOST=stby-scan)(PORT = 1521)))
(CONNECT_DATA =
(SERVER = DEDICATED)
(SERVICE_NAME = reports)
)
)
```

Didn't find what you are looking for?