

Best Practices for Cloning ASM-managed Databases

An Oracle White Paper
September 2006

Best Practices for Cloning ASM-managed Databases

Introduction	3
Using RMAN to Duplicate Databases in ASM Environments	4
Duplicate the database within the same ASM cluster (from one disk group to another)	4
Duplicate the database to a remote server	7
Creating Clone Databases using DBMS_FILE_TRANSFER	9
Overview	9
Preparation	9
Clone Database Creation Procedure	13
Cleanup Procedure	15
Creating Clone Databases using Enterprise Manager	16
Conclusion	16
References	17
Appendix A SET NEWNAME Command Generation Script	18
Appendix B Sample init.ora for Mover Database	19
Appendix C Standby Database Creation Procedures Using DBMS_FILE_TRANSFER Method	20
Preparation	20
Clone Database Creation Procedure	20
Appendix D Sample init.ora for Primary and Standby Databases	23

Best Practices for Cloning ASM-managed Databases

INTRODUCTION

Cloning and refreshing databases for development, testing, and reporting purposes is a common activity for most DBAs. The methods used can range from third party tools to OS or storage-level procedures, and can require additional licensing costs as well as specialized expertise in hardware and storage technologies. As DBAs know their data the best, DBAs should be fully empowered to clone the database for their own needs. By utilizing Oracle's built-in set of cloning tools, available in both command-line and GUI, DBAs can do just that, with no additional software costs and full support from Oracle.

With the introduction of Automatic Storage Management (ASM) in Oracle Database 10g, many databases are utilizing ASM to provide the database administrator with a simple storage management interface that is consistent across all server and storage platforms. As a vertically integrated filesystem and volume manager, purpose-built for Oracle database files, ASM provides the performance of async I/O with the easy management of a filesystem.

This paper presents three techniques for cloning ASM-managed databases:

- [Oracle Recovery Manager \(RMAN\) DUPLICATE command](#). This command automates the database cloning process using RMAN. RMAN ensures that all files needed by the clone database are backed up and copied to the new clone location.
- [DBMS_FILE_TRANSFER package](#). This method transfers all needed files directly from the ASM cluster on the source database to the ASM cluster on the clone database using a 'mover' instance at the clone database server.
- [Enterprise Manager's Clone Database feature or Database Creation Assistant \(DBCA\)](#). The Clone Database feature automates the creation of clone databases with a wizard to create new databases from filesystem to ASM, ASM to ASM, or ASM to filesystem.

Note: The cloning methods described in this paper can also be used to create standby databases. Refer to the Oracle Data Guard documentation on using the RMAN DUPLICATE FOR STANDBY method.

1. Take RMAN image copy backup to +DISKB
2. Redirect clone database control file to new image copies on +DISKB using RMAN SET NEWNAME and DUPLICATE. Note that DUPLICATE, in this case, will not try to restore files from a backup -- the clone database will just use the new image copies on +DISKB as its own data files.

The detailed procedure is:

1. Copy init.ora on source database to clone database filesystem as `init<sid>.ora`, e.g. `initB.ora`.
2. Make changes to clone database's `initorc11.ora`.

Note: These changes are only specific to the DUPLICATE procedure. Other changes may be needed for the clone database, e.g. archived log destinations.

- Set `control_files` parameter to either a complete pathname or disk group name, e.g.

```
*.control_files='+data/orcl1/controlfiles/control01.ctl'
```

or

```
*.control_files='+data'
```

Note: In the second case, a system-generated control file name will be generated by OMF during DUPLICATE, in the '+data/orcl1/controlfiles' directory, e.g. `Current.289.585757677`.

- Set `db_create_file_dest` parameter to appropriate disk group where files will be duplicated, e.g.

```
*.db_create_file_dest='+data'
```

Note: If clone database files are on non-ASM filesystem, pathnames can just be standard directory pathnames, e.g.

```
*.db_create_file_dest='/u02/orcl1'
```

- If the database version is *10.1.0.5 or higher*, you can choose not to set `db_create_file_dest`, but instead set the `db_file_name_convert` and `log_file_name_convert` parameters, if you require more control over disk group naming. In the following example, each data file in the 'data' source disk group will be stored in +data_clone destination disk group, as will each redo log file in +log source disk group be stored in +log_clone destination disk group:

```
*.db_file_name_convert='+data','+data_clone'  
*.log_file_name_convert='+log','+log_clone'
```

Note: As best practice, only use disk group names for the convert parameter values (e.g. '+data', '+data_clone'). Using full directory pathnames for convert values may ultimately yield a different destination disk group pathname due to OMF.

Note: Use either `db_create_file_dest` or the convert parameters, but do not specify both.

- Change the value of the `DB_NAME` initialization parameter, e.g. `orcl1`.

3. Backup source database, full database image copy to disk group +DISKB

```
> rman
```

```
RMAN> connect target sys/passwd@A;
```

```
RMAN> backup as copy database tag 'clonecopy'  
format '+DISKB/%d/datafile_%f.dbf';
```

```
RMAN> sql 'alter system archive log current';
```

At this point, we have created an image copy of all data files using format `+DISKB/<db_name>/datafile_<fileno>.dbf`.

4. Startup clone instance in NOMOUNT mode

5. Duplicate the source database to clone database

```
RMAN> connect target sys/passwd@A;
```

```
RMAN> connect auxiliary sys/passwd@B;
```

```
RMAN> run {
```

```
set newname for datafile 1 to  
'+DISKB/<db_name>/datafile_1.dbf';
```

```
set newname for datafile 2 to  
'+DISKB/<db_name>/datafile_2.dbf';
```

```
<repeat 'set newname' for all datafiles>1
```

```
...
```

```
duplicate target database to 'B'  
pfile='initB.ora';
```

```
}
```

¹ Refer to Appendix A for SQL that can be used to generate the SET NEWNAME commands for all data files.

Because SET NEWNAME simply points each data file to the image copy in +DISKB (clone database's disk group), the duplicate will not actually restore files, since they are already present. The image copy itself is now used by database B. These changes are made in the clone database control file.

Duplicate the database to a remote server

A diagram of the overall DUPLICATE procedure is shown below. DUPLICATE automates the steps of creating a new control file for the clone database, restoring and recovering a backup of the database, generating a new DBID for the clone database, and finally opening the database for general usage.

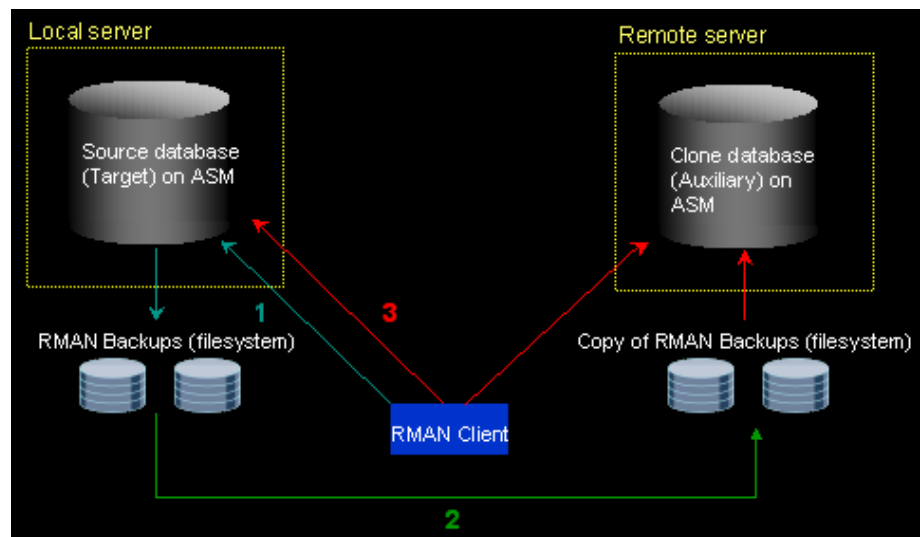


Figure 2 Duplicate Procedure from Local to Remote Server

The overall procedure, as shown in Figure 2, is:

1. Take full RMAN backup to local filesystem.
2. Copy backups to remote server filesystem, keeping the same directories and pathnames. Alternatively, make the local filesystem accessible to the remote server, e.g. via NFS.
3. Perform RMAN DUPLICATE, in which the backups are restored to the ASM-managed clone database on the remote server.

The detailed procedure is listed below. The example syntax in the following steps assumes that source database SID is `orcl` and clone database SID is `orcl1`.

1. Install Oracle database on remote server
 - Setup appropriate ASM disk groups on remote server, including configuration of the ASM cluster password file.

- Copy source database pfile to remote server filesystem, as init<SID>.ora, e.g. initorcl1.ora²
- Verify that remote server can connect to source database via SQLNet, e.g.

```
➤ sqlplus <orcl SYSDBA user>/<orcl
  SYSDBA pwd>@orcl
```

2. Set all needed clone database pfile parameters. Refer to [Step 2 in “Duplicate database within same ASM Cluster”](#) for details about setting these parameters.

3. Take full backup of database to filesystem

```
RMAN> backup as copy format='/u01/backups/%U'
  database plus archivelog;
```

4. Copy backup files to clone host filesystem with same directory structure as the backup files on the primary host (in this example, '/u01/backups/'). Alternatively, make all backup files network-accessible to remote server (e.g. NFS, CIFS).
5. On clone host, startup clone database instance in NOMOUNT mode.
6. On clone host, connect to target database and auxiliary database.

```
➤ > rman target <orcl SYSDBA user>/<orcl SYSDBA
  pwd>@orcl auxiliary <orcl1 SYSDBA
  user>/<orcl1 SYSDBA pwd>@orcl1
```

➤ To duplicate for clone database:

- o RMAN> duplicate target database to 'orcl1' [until sequence <log sequence number>] pfile='initorcl1.ora';
- o Note: The PFILE (initorcl1.ora) must be accessible from remote server.
- o Note: Use until sequence for incomplete recovery, in case the archive log backups cannot satisfy complete recovery of clone database to the current time. Since only backups are made available to the remote server in this procedure, DUPLICATE can only use backups that were taken at or before the requested point-in-time.

Once duplicate finishes successfully, the clone database is opened for use.

² Using an SPFILE for duplicating ASM-managed databases is currently not supported. A PFILE must be used and can be generated from an existing SPFILE if needed.

If the `control_files` parameter in the clone database pfile was specified as a diskgroup name, replace this parameter with the complete pathname for the newly created control file. The newly created control file pathname can be found using:

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

CREATING CLONE DATABASES USING DBMS_FILE_TRANSFER

Overview

This procedure clones the source database by transferring files directly to the ASM cluster of the clone database, using the `DBMS_FILE_TRANSFER` package. In addition to the source and clone databases, an additional ‘mover’ database is required. This mover database is a temporary seed database that resides on the clone database server, and is used solely to assist the transfer of the source database files to the clone server.

Note: If creating standby databases, ensure that [Appendix C](#) is reviewed in its entirety for all procedure changes.

Preparation

The following steps describe the steps to prepare the source database, mover database, and the clone database.

Source Database Setup

- Create directory structures on the source database. This directory should point to the name of ASM diskgroup where the data files and control files reside. If you have more than one diskgroup that contains datafiles, then create the additional directories.

```
Sql> create or replace directory  
SOURCE_DIR_PRDB as  
' +DATA/PRDB_CHICAGO/datafile' ;
```

```
Sql> create or replace directory  
SOURCE_CTL_DIR_PRDB as  
' +DATA/PRDB_CHICAGO/controlfile' ;
```

- Create a clone control file:

```
Sql> alter database backup controlfile to  
' +DATA/PRDB_CHICAGO/controlfile/clone_PRDB.ctl  
' ;
```
- Modify `tnsnames.ora` to add an alias for the clone database.
- On source database, create the following stored procedure to enable the tablespaces to be put in “hot backup” mode dynamically within the file transfer loop script. Note that the stored procedure must be owned by SYS.

```

CREATE or replace PROCEDURE alter_tblspce
(tablespace_name IN VARCHAR2, new_state in
varchar2) AS
    cid INTEGER;
BEGIN
    -- open new cursor and return cursor ID
    cid := DBMS_SQL.OPEN_CURSOR;
    -- parse and immediately execute dynamic
SQL statement
--built by concatenating tablespace name to
the alter
--end backup command
    DBMS_SQL.PARSE(cid, 'alter tablespace ' ||
tablespace_name || ' ' || new_state || ' backup
', dbms_sql.v7);
    -- close cursor
    DBMS_SQL.CLOSE_CURSOR(cid);
EXCEPTION
    -- if an exception is raised, close cursor
before exiting
    WHEN OTHERS THEN
        DBMS_SQL.CLOSE_CURSOR(cid);
        -- reraise the exception
        RAISE;
End alter_tblspce;
/

grant execute on alter_tblspce to system;

```

Mover Database Setup (on Clone Server)

- Create ASM diskgroups equal in size to those on the source database server. **Use the same diskgroup names as the source database.**
- Using DBCA (or any other method), create the mover database on the clone server with the ASM diskgroup(s) specified above. [Appendix B](#) has a sample `init.ora` that can be used for the mover database. Make sure an `SPFILE` is created for the mover database. If DBCA was, then `SPFILE` will be created by default. The mover database can be named *mover* if desired.
- Create a database link on the mover database to connect to the source database. Note that you must also update the `tnsnames.ora` file to include the alias.

```
Sql>create database link PRDB_CHICAGO
      CONNECT TO SYSTEM IDENTIFIED BY
PRDB_CHICAGO
      USING 'PRDB_CHICAGO'
```

- On the clone server's ASM instance, create an ASM directory to hold the aliases for the files to be copied over.

```
Sql> alter diskgroup DATA add directory
'+DATA/primary_files_PRDB';
```

- Create a database directory on the mover database to reflect the ASM directory created above. This directory will house the PRDB database files.

```
Sql>create or replace directory CLONE_DIR_PRDB
as '+DATA/primary_files_PRDB';
```

- Create a database directory on mover database to reflect the ASM directory that will house the clone control file.

```
Sql>create or replace directory
CLONE_CTL_DIR_PRDB as
'+DATA/PRDB_BOSTON/controlfile';
```

Clone Database Setup

- Setup the clone database directory structures for `bdump`, `cdump`, `udump`, and `pfile`.
- Create a password file for the clone instance.
- For clone database creation, the clone database `init.ora` should be set appropriately for `db_name`, `db_create_file_dest`, `db_recovery_file_dest`, and `control_files`. The `control_files` parameter should be set to the location of the to-be-transferred control file, i.e.
`'+DATA/PRDB_BOSTON/controlfile/clone_PRDB.ctl'`.
- Create an SPFILE from the `init.ora`.

Clone Database Creation Procedure

File Transfer

1. Ensure that source database is active.
2. After connecting to mover database via SQL*Plus, use DBMS_FILE_TRANSFER to transfer the clone control file that was created on the primary node.

```
conn / as sysdba;
begin
    dbms_file_transfer.get_file(
        source_directory_object =>
            'SOURCE_CTL_DIR_PRDB',
        source_file_name         =>
            'clone_PRDB.ctl',
        destination_directory_object =>
            'CLONE_CTL_DIR_PRDB',
        destination_file_name     =>
            'clone_PRDB.ctl',
        source_database           => 'PRDB_CHICAGO' );
END;
/
```

3. While connected to the mover database, transfer the source database files using the DBMS_FILE_TRANSFER package. Note, the source database tablespaces must be in hot backup mode during the copy. The following procedure will loop through all the tablespaces, placing them in “hot backup” mode, copy the files to the mover database, and end the hot backup mode.

```

set serveroutput on
declare cursor get_primary_files_cur is
select substr(name,instr(name,'/',-1)+1),
substr(substr(name,1,instr(name,'.',-1,2)-
1),instr(name,'/',-1)+1)
from v$datafile@PRDB_CHICAGO;
primary_file_name varchar2(513);
tspace_name       varchar2(513);
begin
open get_primary_files_cur;
loop
  fetch get_primary_files_cur into primary_file_name,
  tspace_name;
  exit when get_primary_files_cur%NOTFOUND;
  sys.alter_tbspce@PRDB_CHICAGO(
    tablespace_name => TSPACE_NAME,
    new_state       => 'BEGIN');
  dbms_output.put_line('BEGIN TRANSFER OF TABLESPACE '
|| TSPACE_NAME);
  dbms_file_transfer.get_file(
    source_directory_object => 'SOURCE_DIR_PRDB',
    source_file_name        => PRIMARY_FILE_NAME,
    destination_directory_object => 'CLONE_DIR_PRDB',
    destination_file_name    => TSPACE_NAME,
    source_database         => 'PRDB_CHICAGO');
  sys.alter_tbspce@PRDB_CHICAGO(
    tablespace_name => TSPACE_NAME,
    new_state       => 'END');
  dbms_output.put_line('END TRANSFER OF TABLESPACE '
|| TSPACE_NAME);
end loop;
end;
/

```

4. Ensure all files have been transferred to the clone server.

5. Shutdown the mover database.

Enabling the Clone Database

1. Ensure that all needed values are set in the clone database `init.ora` file.
2. Transfer all archived redo logs, created after the database was put in hot backup mode, to the clone server. To transfer the archived redo logs, leverage the same method used for transferring the datafiles.
3. Startup the clone database in nomount mode.

```
Sql> startup nomount
```

4. Using RMAN, connect to the clone database, and issue the `catalog` command. This command updates the clone database control file with the new file/directory structure.

```
rman target system/manager1
rman> catalog start with '+DATA/primary_files_PRDB';
```

4. You will be prompted to catalog the files that were found; i.e., *unknown* to the clone database. At this prompt enter YES to catalog all the unknown files to the clone database. Upon completion, all cataloged files will be displayed for verification³.

5. Now allow the clone database to maintain ownership of these database files.

```
rman>switch database to copy;
```

6. Recover the clone database and open with resetlogs.

```
SQL> RECOVER DATABASE;  
SQL> ALTER DATABASE OPEN RESETLOGS;
```

7. Finally, change the clone database DBID.

Shutdown consistently and startup in mount:

```
SQL> SHUTDOWN IMMEDIATE  
SQL> STARTUP MOUNT
```

Invoke DBNEWID utility, e.g.:

```
> nid TARGET=SYS/oracle@PRDB_BOSTON
```

Open database:

```
Sql> ALTER DATABASE OPEN RESETLOGS;
```

Cleanup Procedure

1. Drop the aliases created on the mover database. This needs to be done on the ASM instance on the secondary server.

```
Sql> alter diskgroup DATA drop directory  
      '+DATA/primary_files_PRDB force;
```

2. Drop the clone control file on the source database, e.g.

```
Sql> alter diskgroup DATA drop file  
      '+DATA/PRDB_CHICAGO/controlfile/clone_PRDB.ctl';
```

3. Shutdown the mover database, if not already shutdown. Note, the mover database can be used again for future database cloning.

³ Currently in Oracle Database 10gR2, the file names to be cataloged begin with the following format `FILE_TRANSFER_fileno.incarnation_n`. The alias names in `+DATA/primary_files_PRDB` will not be cataloged. Note that this naming format does not interfere with normal operations of the clone database.

CREATING CLONE DATABASES USING ENTERPRISE MANAGER

EM Database Control supports ASM-managed database cloning on the same server through the Clone Database feature. If cloning needs to be performed across servers, then EM Grid Control is required.

With the Clone Database feature, there only needs to be enough staging area space on the source and destination server for the largest data file, as the source database files are transferred via HTTP or FTP and restored to the clone database one at a time. If the servers can share an NFS mount, shared drive, or if the cloning is performed on the same server, only one staging area is required. Another option is to clone the database using a full backup of the database, which requires a larger staging area, and optionally saving this staging area for future cloning operations. In addition, if using Grid Control and a saved backup in the staging area, no connection to the source database is required for the cloning process. More information on the Clone Database feature can be found in EM Online Help.

The Database Creation Assistant (DBCA) can also be used to create ASM-managed databases. With DBCA, a staging area for the entire database is first created, and cloning can then proceed without requiring connection to the source database. The staging area must be locally accessible by the destination server. For more information on DBCA, refer to the [Oracle Database 2-day DBA documentation on DBCA](#) and [creating DBCA templates for cloning databases from an existing database](#).

EM Grid Control supports ASM-managed standby database creation via the Add Standby Database wizard on the Data Guard Overview page. For more information, refer to [Chapter 6 “Scenarios Using Oracle Enterprise Manager” in the Data Guard Broker](#) documentation.

CONCLUSION

Several options exist to clone ASM-managed databases:

- RMAN DUPLICATE takes advantage of an existing backup to clone the database to another server.
- DBMS_FILE_TRANSFER package can be used to transfer the files directly from the source to the destination database.
- For EM Grid Control users, the Clone Database feature can be used to clone a database from one server to another, and only requires temporary staging area space on each server equal to the largest datafile. Another option is to clone from a previously saved staging area, accessible to the destination server; this does not require connection to the source database.
- DBCA is a standalone tool that can be used to create clone databases from a previously created ‘clone staging area’, with the option of cloning just the database structure, or structure and data. No connection to the source database is required.

REFERENCES

Administrator's Guide, [Using Automatic Storage Management \(ASM\)](#)

Backup and Recovery Advanced User's Guide, [Creating and Updating Duplicate Databases with RMAN](#)

APPENDIX A SET NEWNAME COMMAND GENERATION SCRIPT

The following SQL at the source database will output a list of needed SET NEWNAME commands:

```
SQL> set heading off
SQL> set feedback off
SQL> set sqlprompt " "
SQL> set linesize 1000
SQL> select 'set newname for datafile ' || file# || '
to ''' || name || ''''
from v$datafile_copy where status = 'A' and tag =
'clonecopy' order by file#;
```

Note: tag='clonecopy' was the tag assigned to the image copy backup.

Ensure that the output is formatted properly (e.g. remove banner and column names), before copying into the run {..} script.

APPENDIX B SAMPLE INIT.ORA FOR MOVER DATABASE

This section illustrates sample init.ora parameter file used to support the Mover database.

Mover database init.ora:

```
*.background_dump_dest='/opt/oracle/product/10.1.0.3/admin/PRDB/bdump'
*.compatible='10.1.0.2.0'
*.control_files='+DATA/PRDB/controlfile/current.256.1','+FLASH/PRDB/controlfile/current.256.1'
*.core_dump_dest='/opt/oracle/product/10.1.0.3/admin/PRDB/cdump'
*.db_block_size=8192
*.db_cache_size=25165824
*.db_create_file_dest='+DATA'
*.db_domain=''
*.db_file_multiblock_read_count=16
*.db_name='MOVER'
*.db_unique_name='PRDB_BOSTON'
*.db_recovery_file_dest_size=64424509
*.db_recovery_file_dest='+FLASH'
*.java_pool_size=50331600
*.job_queue_processes=10
*.large_pool_size=8388608
*.open_cursors=300
*.pga_aggregate_target=25165824
*.processes=250
*.remote_login_passwordfile='exclusive'
*.shared_pool_size=99614720
*.sort_area_size=65536
*.undo_management='AUTO'
*.user_dump_dest='/opt/oracle/product/10.1.0.3/admin/PRDB/udump'
```

APPENDIX C STANDBY DATABASE CREATION PROCEDURES USING DBMS_FILE_TRANSFER METHOD

The following procedure changes are specific to creating standby databases using the DBMS_FILE_TRANSFER method.

Preparation

Source Database Setup

- Instead of creating a clone control file, create a standby control file:

```
Sql> alter database create standby controlfile as  
' +DATA/PRDB_CHICAGO/controlfile/clone_PRDB.ctl';
```

- Specify the Data Guard configuration file location. This Data Guard metadata file is used to record the last known valid state of the configuration. For more information on this file, refer to the Oracle Data Guard Broker documentation.

Mover Database Setup (on Clone Server)

- If creating a standby database, the `init.ora` parameter `db_unique_name` for the mover database must be set to the **same** name as the standby database. Because of this, the mover and standby database cannot be active at the same time.

Clone Database Setup

- When creating a password file for the standby database, copy the password file from primary to the secondary server. This ensures that the SYS user password in the password file is the same as the one on the primary database.
- Update the standby database `init.ora` file with the correct `LOG_ARCHIVE_DEST_N` and control file location. In the case of the control file setting, it should be set to the location of the standby control file, i.e.
`' +DATA/PRDB_BOSTON/controlfile/clone_PRDB.ctl'.`
- Refer to [Appendix D](#) for other parameters to set in a standby database `init.ora`.

Clone Database Creation Procedure

Enabling the Clone Database

1. Startup the standby database.

```
Sql> startup nomount
```

```
Sql>alter database mount standby database
```

2. On the standby database, set the following parameters to enable automatic standby file management. This allows file management operations such as adding and deleting files to be done automatically by Oracle on the standby database.

```
Sql>Alter system set STANDBY_FILE_MANAGEMENT=AUTO  
scope=both sid='*';
```

```
Sql>Alter system set db_recovery_file_dest='+FLASH'  
scope=both sid='*';
```

```
Sql>Alter system set db_create_file_dest='+DATA'  
scope=both sid='*';
```

3. Specify the Data Guard configuration file location. This Data Guard metadata file is used to record the last known valid state of the configuration. For more information on this file, refer to the Oracle Data Guard Broker documentation.
4. Verify that Data Guard FAL mechanism is sending the archived logs from primary to standby database. Make sure primary database `init.ora` file is using the correct `LOG_ARCHIVE_DEST_2`, which points to the standby database.
5. Verify the archive transmission and status using the following SQL on primary database and standby database.

```
Sql>select dest_name, status, type, destination  
from v$archive_dest_status;
```

6. Once all datafiles are transferred to the secondary server, they will need to be cataloged with the standby database (not the mover database). Startup the standby database in mount mode.
7. Using RMAN, connect to the standby database, and issue the `CATALOG` command. This process essentially updates the new standby database control file with the new file/directory structure.

```
rman target system/manager1
```

```
rman> catalog start with '+DATA/primary_files_PRDB';
```

You will be prompted to catalog the files that were found; i.e., *unknown* to the standby database. At this prompt enter YES to catalog all the unknown files to the standby database. Upon completion, all cataloged files will be displayed for verification⁴.

⁴ Currently in Oracle Database 10gR2, the file names to be cataloged begin with the following format `FILE_TRANSFER_fileno.incarnation_no`. The alias names in

8. Now allow the standby database to maintain ownership of these database files.

```
rman> switch database to copy;
```

9. Begin managed recovery.

```
SQL> RECOVER MANAGED STANDBY DATABASE NODELAY  
DISCONNECT;
```

10. Verify archived logs are being used. On primary database, perform the following:

```
Sql>alter system switch logfile;  
Sql>select * from v$archive_dest;
```

Also, verify that the archived logs have been transmitted to the standby database.

+DATA/primary_files_PRDB will not be cataloged. Note that this naming format does not interfere with normal operations of the standby database.

APPENDIX D SAMPLE INIT.ORA FOR PRIMARY AND STANDBY DATABASES

This section illustrates sample `init.ora` parameter files for the primary and standby databases.

Chicago (Primary Database)

```
*.FAL_CLIENT='PRDB_CHICAGO'
*.FAL_SERVER='PRDB_BOSTON'
*.DB_UNIQUE_NAME='PRDB_CHICAGO'
*.LOG_ARCHIVE_CONFIG='DG_CONFIG=
(PRDB_CHICAGO,PRDB_BOSTON)
*.STANDBY_ARCHIVE_DEST=
USE_DB_RECOVERY_FILE_DEST
*.LOG_ARCHIVE_DEST_1='location=
USE_DB_RECOVERY_FILE_DEST arch noreopen max_failure=0 mandatory
valid_for=(ALL_LOGFILES,ALL_ROLES) db_unique_name=PRDB_CHICAGO'
*.LOG_ARCHIVE_DEST_2='service=
PRDB_BOSTON reopen=15 max_failure=10 lgwr affirm
valid_for=(ONLINE_LOGFILES,PRIMARY_ROLE) db_unique_name=PRDB_BOSTON'
*.db_recovery_file_dest='+FLASH'
*.db_recovery_file_dest_size=64424509
```

Boston (Physical Standby Database)

```
*FAL_CLIENT='PRDB_BOSTON'
*.FAL_SERVER='PRDB_CHICAGO'
*.DB_UNIQUE_NAME='PRDB_BOSTON'
*.LOG_ARCHIVE_CONFIG='DG_CONFIG=
(PRDB_CHICAGO,PRDB_BOSTON)
*.STANDBY_ARCHIVE_DEST=
USE_DB_RECOVERY_FILE_DEST='+FLASH'
*.LOG_ARCHIVE_DEST_1='location=
USE_DB_RECOVERY_FILE_DEST arch noreopen max_failure=0
mandatory valid_for=(ALL_LOGFILES,ALL_ROLES)
db_unique_name=PRDB_BOSTON'
*.LOG_ARCHIVE_DEST_2='service=PRDB_CHICAGO
reopen=15 max_failure=10 lgwr affirm
valid_for=(ONLINE_LOGFILES,PRIMARY_ROLE)
db_unique_name=PRDB_CHICAGO'
*.db_recovery_file_dest='+FLASH'
*.db_recovery_file_dest_size=64424509
```



Best Practices for Cloning ASM-managed Databases

September 2006

Authors: Timothy Chien, Nitin Vengurlekar, Muthu Olagappan

Contributing Authors: Tammy Bednar

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com

Copyright © 2005, Oracle. All rights reserved.

This document is provided for information purposes only and the contents hereof are subject to change without notice.

This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission. Oracle, JD Edwards, and PeopleSoft are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.