Cloning the AZDBA01 Staging Database

The following outlines the steps followed in order to clone, using RMAN active duplicate, the AZDBA01 database, to the other required databases on the three Azure servers ORCDEVORC01 through 03.

*You cannot clone a running database to a new database, with the same name, regardless of whether it is running on a different server or not.*

# Terminology:

* target\_database. The source database. (Oracle refer to this as the target, go figure!)
* target\_server. Where the target database runs.
* auxiliary\_database. The destination database. (Oracle refer to this as the auxiliary database.)
* auxiliary\_server. Where the auxiliary database runs.

# Prepare the target server

* Start the target\_database using an spfile. This can be checked by:

sqlplus / as sysdba

show parameter spfile

There *must* be a valid spfile name returned. If not, the database must be restarted using an spfile.

create spfile=’?\database\spfile<databasename>.ora’ from pfile;

startup force pfile=’?\database\spfile<databasename>.ora’

* This database must be running in archivelog mode. This can be checked by:

select log\_mode from v$database;

* If returned value is NOARCHIVELOG then run the following:
  + shutdown immediate;
  + startup mount;
  + alter database archivelog;
  + alter database open;
  + alter system archive log current;
  + archive log list
* Make sure that the tnsnames.ora has an entry for the auxiliary\_database. This can be checked by running:

tnsping auxiliary\_database

There should be an ok status returned at the end of the output. If not, add the auxiliary\_database to the target\_server's tnsnames.ora file.

# Prepare the auxiliary server

* Open an Administrator enabled command session. Run the following command replacing "<auxiliary\_database>" as appropriate:

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

* Create the folder structure required by the auxiliary\_database. For example, run the following in a cmd session to easily create the full paths:

mkdir <drive>:\mnt\oradata\azppd01

mkdir <drive>:\mnt\fast\_recovery\_area\azppd01

* Copy the password file for the target\_database, from %ORACLE\_HOME%\database\pwd<target\_database>.ora on the target\_server to %ORACLE\_HOME%\database\pwd<auxiliary\_database>.ora on the auxiliary\_server.
* Create a new pfile for the auxiliary database. It only needs to contain the following:

db\_name=<auxiliary\_database name>

* Save the file in %ORACLE\_HOME\DATABASE\init<auxiliary\_database name>.ora
* Add an entry for the auxiliary\_database to the auxiliary\_server's tnsnames.ora and also to the primary\_server's tnsnames.ora.
* Add an entry for the database to the listener.ora.
* Stop and start the listener service.
* Start the auxiliary\_database in NOMOUNT mode. It must be started using a pfile, not an spfile.

set oracle\_home=C:\OracleDatabase\product\11.2.0\dbhome\_1

set nls\_lang=American\_america.we8iso8859p1

set nls\_date\_format=

set oracle\_sid=<auxiliary\_database name>

sqlplus sys/<sys\_password> as sysdba

startup nomount pfile=’?\DATABASE\init<auxiliary\_database name>.ora’

# Clone the database

The database is ready to be cloned. It can be initiated from the target\_server or from the auxiliary\_server as desired.

* Connect to RMAN using a password for both the target and auxiliary databases. There must also be a tnsnames.ora alias used for the auxiliary database. For best results, use one on both databases:

rman target sys/password@target\_database auxiliary sys/password@auxiliary\_database

* If you are cloning onto different servers, see the [instructions here](#_Cloning_A_Staging_3).
* If you are cloning onto the same server, see [these instructions](#_Cloning_A_Staging_2) instead.

Don't forget to return here after the clone has completed. By the way, ignore any errors relating to Block Change Tracking files. We fix those next.

# Post Cloning Tidy Up and Checks

After the clone has finished it is wise to make sure everything is in order. Cloning a database in this manner will, for example, change the instance\_name parameter, amongst others, back to that of the target\_database even though the auxiliary\_database was started with its own pfile.

The first step is to fix the block change tracking problem. You may have seen a message similar to the following:

ORA-19750: change tracking file: 'G:\MNT\FAST\_RECOVERY\_AREA\AZAZDBA01\DCT.DBF'

ORA-27040: file create error, unable to create file

OSD-04002: unable to open file

O/S-Error: (OS 3) The system cannot find the path specified.

This is fixed as follows, and should be executed on the auxiliary database:

select status, filename from v$block\_change\_tracking;

If everything looks ok, then good, otherwise:

alter database enable block change tracking

using file 'path\to\fra\for\AZPPD01\bct.dbf';

The following SQL can be used on the clone database to identify initialisation parameters that may need adjusting:

select name, value

from v$parameter

where upper(value) like '%XXXXXX%'

and lower(name) not like '%file\_name\_convert';

Where 'XXXXXX' is the staging database's name in upper case. For example if the database was cloned from AZDBA01:

select name, value

from v$parameter

where upper(value) like '%AZDBA01%'

and lower(name) not like '%file\_name\_convert';

The results *might* look as follows:

dispatchers

(PROTOCOL=TCP) (SERVICE=AZDBA01XDB)

instance\_name

AZDBA01

service\_names

AZDBA01

To resolve the issues identified above, run the following SQL, if necessary:

alter system set instance\_name='AZPPD01' scope=spfile;

alter system set service\_names='AZPPD01' scope=spfile;

alter system set audit\_file\_dest = 'C:\ORACLEDATABASE\ADMIN\AZPPD01\ADUMP' scope = spfile;

alter system set dispatchers=

'(PROTOCOL=TCP) (SERVICE=AZPPD01XDB)' scope=spfile;

If the database is to continue to run in ARCHIVELOG mode, then simply:

alter database flashback on;

startup force

And the check SQL run again. This time, there should be no rows selected.

If the database is to be run in NOARCHIVELOG mode, then:

startup force mount

alter database flashback off;

alter database noarchivelog;

alter database open;

Two roles will now require to be updated as their password is dependent on the database name, so they currently have the password of the originating database:

column db\_name new\_value my\_dbname noprint;

select name as db\_name from v$database;

alter role NORMAL\_USER identified by &&my\_dbname.123;

alter role SVC\_AURA\_SERV\_ROLE identified by &&my\_dbname.123;

Check that dba\_scheduler\_jobs are disabled:

select owner, enabled, job\_name

from dba\_scheduler\_jobs

where enabled = 'TRUE'

and owner not in ('SYS','SYSTEM','SYSMAN','ORACLE\_OCM','EXFSYS')

order by owner,job\_name;

For a non-production database, there should be no jobs owned by FCS in the listing. If there are, they must be disabled:

dbms\_scheduler.disable(name => 'whatever');

**Configuration of Cloned Databases**

After cloning any non-production *depersonalised* databases, we must run the following script – you may ignore any errors relating to dropping of objects. The script in question *must be run as the FCS user*, and is located in TFS at:

$/TA/MAIN/Source/UKRegulated/Database/Depersonalisation/Depers & Shrink/8\_uat\_config.sql

connect fcs/password

@8\_uat\_config.sql

There are also various user creation scripts which can be found in TFS at location:

$/TA/MAIN/Non Source/Dev DBA/Database Release/control\_script/Create\_UV\_Users/Main

The controlling script is named execute.sql and this *must* be edited prior to *running as the FCS user*. Only one line needs to be changed:

PK\_ACCESS\_SETUP.UPDATE\_ACCESS('UAT');

Replace 'UAT' above with one of the other values listed in the file itself. The value depends on the "type" of the database. Save and run the code:

sqlplus fcs/password @execute.sql

If you mistakenly run the code as SYS, then the fix is:

drop package pk\_access\_setup;

connect FCS/password

@pk\_access\_setup\_pks.sql

@pk\_access\_setup\_pkb.sql

declare

vout varchar2(100);

begin

-- Possible values

-- DEMO, TRG, DEV, ST, SIT, UAT, PPROD, LIVE

PK\_ACCESS\_SETUP.UPDATE\_ACCESS('UAT');

end;

/

DROP PACKAGE pk\_access\_setup;

# Register Database for RMAN Backups

If the databases are to be backed up using RMAN, then they must be registered with the RMAN catalog.

rman target / catalog rman11g/rman11gcatalog@azrmn01

register database;

run {

configure controlfile autobackup on;

configure backup optimization on;

configure retention policy to recovery window of 7 days;

configure archivelog deletion policy to backed up 2 times to disk;

}

show all;

exit

# Cloning A Staging Database to a Different Server

The following outlines the steps followed in order to clone, using RMAN active duplicate, the AZDBA01 database, to a new database named azppd01 on the Azure server ORCDEVORC03. Exactly the same directory structure was used on the auxiliary\_server as on the target\_server. This need not always be the case, however.

* Run the following command, replacing azppd-01 with the auxiliary\_database's name as appropriate:

run {

allocate auxiliary channel x1 device type DISK;

allocate auxiliary channel x2 device type DISK;

allocate auxiliary channel x3 device type DISK;

allocate channel d1 device type DISK;

allocate channel d2 device type DISK;

allocate channel d3 device type DISK;

allocate channel d4 device type DISK;

allocate channel d5 device type DISK;

duplicate target database to azppd01

from active database

spfile

parameter\_value\_convert

'g:\mnt\oradata\azdba01',

'g:\mnt\oradata\azppd01',

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

'g:\mnt\fast\_recovery\_area\azppd01'

set instance\_name 'azppd01'

set service\_names 'azppd01'

set dispatchers '(PROTOCOL=TCP) (SERVICE=AZPPD01XDB)'

set audit\_file\_dest 'C:\ORACLEDATABASE\ADMIN\AZPPD01\ADUMP'

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

set dg\_broker\_start 'false'

set control\_files

'g:\mnt\oradata\azppd01\control01.ctl',

'g:\mnt\fast\_recovery\_area\azppd01\control02.ctl'

set db\_file\_name\_convert

'g:\mnt\oradata\azdba01',

'g:\mnt\oradata\azppd01',

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

'g:\mnt\fast\_recovery\_area\azppd01'

set log\_file\_name\_convert

'g:\mnt\oradata\azdba01',

'g:\mnt\oradata\azppd01',

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

'g:\mnt\fast\_recovery\_area\azppd01'

nofilenamecheck;

release channel x1;

release channel x2;

release channel x3;

release channel d1;

release channel d2;

release channel d3;

release channel d4;

release channel d5;

}

**Warning:** The PARAMETER\_VALUE\_CONVERT is *supposed* to rename the settings for the control\_files etc, but appears not to work (at least on Azure). To this end, it was necessary to recreate the target\_server's tree structure for where the control files lived on the auxiliary\_server too. This also left the control files in the wrong location after the clone.

By specifying the control\_files parameter above, this problem was worked around. See [below](#_Control_Files_Workaround) for a workaround for when this parameter wasn't used.

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

In testing, three auxiliary and three disk channels allowed three files to be restored over the network between servers, in 30 minutes. Each file is 22 Gb in size. This allows for an estimate of approximately 6-8 hours per full database clone. There are 43 data files in a full database. (But they are not all 22gb in size!)

By way of comparison, a full cold backup on the same server only took a shade over 2 hours.

# Cloning A Staging Database to the Same Server

The following outlines the steps followed in order to clone, using RMAN active duplicate, the AZDBA01 database, to a new database on the same Azure server ORCDEVORC01.

* Run the following command, replacing the auxiliary\_database's name as appropriate:

run {

allocate auxiliary channel x1 device type DISK;

allocate auxiliary channel x2 device type DISK;

allocate auxiliary channel x3 device type DISK;

allocate channel d1 device type DISK;

allocate channel d2 device type DISK;

allocate channel d3 device type DISK;

allocate channel d4 device type DISK;

allocate channel d5 device type DISK;

duplicate target database to azxxx01

from active database

spfile

parameter\_value\_convert

'g:\mnt\oradata\azdba01',

'h:\mnt\oradata\azxxx01',

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

'h:\mnt\fast\_recovery\_area\azxxx01'

set instance\_name 'azppd01'

set service\_names 'azppd01'

set dispatchers '(PROTOCOL=TCP) (SERVICE=AZPPD01XDB)'

set audit\_file\_dest ' C:\ORACLEDATABASE\ADMIN\AZPPD01\ADUMP'

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

set dg\_broker\_start 'true'

set control\_files

'h:\mnt\oradata\azxxx01\control01.ctl',

'h:\mnt\fast\_recovery\_area\azxxx01\control02.ctl'

set db\_file\_name\_convert

'g:\mnt\oradata\azdba01',

'h:\mnt\oradata\azxxx01',

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

'h:\mnt\fast\_recovery\_area\azxxx01'

set log\_file\_name\_convert

'g:\mnt\oradata\azdba01',

'h:\mnt\oradata\azxxx01',

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

'h:\mnt\fast\_recovery\_area\azxxx01'

;

release channel x1;

release channel x2;

release channel x3;

release channel d1;

release channel d2;

release channel d3;

release channel d4;

release channel d5;

}

**Warning:** The nofilenamecheck parameter *must* be *omitted*. This prevents the clone process from inadvertently overwriting target\_database files rather than auxiliary\_database files. This parameter *must* *never* be specified when cloning to the same server.

**Warning:** The PARAMETER\_VALUE\_CONVERT is *supposed* to rename the settings for the control\_files etc, but appears not to work (at least on Azure). To this end, it was necessary to recreate the target\_server's tree structure for where the control files lived on the auxiliary\_server too. This also left the control files in the wrong location after the clone.

By specifying the control\_files parameter above, this problem was worked around. See [below](#_Control_Files_Workaround) for a workaround for when this parameter wasn't used.

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

In testing, three auxiliary and three disk channels allowed three files to be restored over the network between servers, in 30 minutes. Each file is 22 Gb in size. This allows for an estimate of approximately 6-8 hours per full database clone. There are 43 data files in a full database. (But they are not all 22gb in size!)

By way of comparison, a full cold backup on the same server only took a shade over 2 hours.

When complete, proceed to tidy up after the clone process. See below.

# Control Files Workaround

As mentioned above, the PARAMETER\_VALUE\_CONVERT parameter in the duplicate command should have renamed the control files appropriately for the cloned database, however, it did not. Without using the control\_files parameter, the target\_database's control files were copied to an exact replica of the target\_server's directory structure on the auxiliary\_server.

This meant that there needed to exist, a structure as follows, on the auxiliary\_server:

g:\mnt\oradata\target\_database

g:\mnt\fast\_recover\_area\target\_database

When what we really wanted was the following:

g:\mnt\oradata\auxiliary\_database

g:\mnt\fast\_recover\_area\auxiliary\_database

To fix the database and put the controlfiles into the correct location, follow the following steps:

* ALTER SYSTEM SET CONTROL\_FILES=

'g:\mnt\oradata\auxiliary\_database\CONTROL01.CTL'.

'g:\mnt\fast\_recover\_area\auxiliary\_database\CONTROL02.CTL'

SCOPE = SPFILE;

Obviously, replace "auxiliary\_database" with the appropriate folder name.

* SHUTDOWN
* In the operating system, after the database *has fully shutdown*, copy the current control files to the locations and names noted above.
* STARTUP

The control files should now be in the correct place as desired and the ones named after the target\_database's locations can be deleted from the auxiliary\_server (only!)