9i RAC: Cloning to a Single Instance

Shankar Govindan

Introduction

Cloning is a process that most DBA's are aware of that needs to be done often to keep the production database from being vandalized by the developers and UAT's. A production database needs to be cloned to test or development or training databases, so that the Application development and any customization can be tested and validated, before porting to production. User acceptance tests are not done and signed off after you go production, but by testing on a copy or clone of production at a point in time. Some sites need to clone often where data integrity and validation is important, whereas others need to be cloned once in few weeks. Either way, cloning is done to test upgrades or when patches are tested on a production copy before applying it to production. So, knowledge of cloning process is a must in any DBA's job profile.

In this paper we are going to look at one of the important administration jobs, the cloning of a 9i RAC. We will look at how we could clone a 9iRAC database which can have any number of nodes (4 is the limit I think on Solaris and depends on the OS and cluster volume managers) to a single instance. When I say a single instance, it is still going to be a single 9iRAC instance for a RAC database and not a non-RAC database. To down grade after a clone, look up the downgrade document on Metalink.

Backup

Cold Backups are not the norm anymore, it has vanished from almost all the sites and hot backup is the right way to backup a database. Export/Import can be used for cloning, but it is more laborious, time consuming and prone to error. Any hot backup process has three basic steps and an additional one to flush the redologs.

- Oracle tablespaces are put in the begin backup mode,
- the database files of the tablespace are copied using operating system commands and
- then the tablespaces are put back to end backup mode.

During this begin backup and end backup phase, the datafiles are written to and when the end backup command is issued, they are brought to a consistent state by the redologs. Since the online redologs have the last database information, these redologs need to be flushed to archivelogs, so a consistent database backup set is available for cloning. Remember, we do not copy over the online redo logs for cloning like we do with a cold backup, hence the backup set is meaningful only if the redologs are flushed after the end backup and these archivelogs are part of the backup set. The datafiles in the backup set will all be out of sync and hence need recovery.

Flushing the log buffer or switching a logfile synchronizes the modified data blocks in memory with the datafiles on disk as a checkpoint occurs; this is done using the command:

```
Alter system switch logfile;
```

But in a single instance, this would help. If there are multiple instances, then each instance will have its own redologs called threads. These redo threads create their own archivelogs in their own directory if the archive_log_dest is set in the init parameter file for specific instances to dump to these directories.

A backup set in a Real Application Cluster database needs all the instances archivelogs to be switched after an end backup. We initiate a backup operation from a single instance, and the

command above will only switch the redologs for a single instance. To switch all the instances in a RAC environment for a meaningful backup set, we need to use the command:

```
Alter system archive log current;
```

This command after a backup is very important in a RAC cloning process; hence make sure this command is part of your backup scripts.

Cloning Process Flow

Cloning of 9i RAC Production server to a Test server was done by copying over the hot backup database files and the required archivelogs. The server names and instance names are:

Production servers are	: ODS044 and ODS045
Test servers are	: ODS034 and ODS035
Production instances are	: SIDP1 and SIDP2
Test Instances are	: SIDQ1 and SIDQ2
Production database	: SIDP
Test database	: SIDQ

Operating System	: Solaris 8
Filesystem	: Veritas with Cluster management.

The basic steps involved in cloning are as below, I will try to expand the steps below by including as much information as required to successfully clone a 9i RAC.

- A consistent hot backup set.
- Operating system with patch level similar to production.
- Oracle server installation or clone with same patchset level.
- Veritas filesystem setup and cluster management software with patch level same as production.
- Environment setup
- Pfile setup
- Clone database and recover.
- Listener and tnsnames.ora setup
- Server control configuration

Backup Set

When we say a backup set, the set includes the hot backup datafiles and the archivelogs that where generated during the backup and the last one flushed after the end backup mode.

Note: If you have a backup procedure like a shadow copy, then you have the advantage of having the online redologs, otherwise most backup sets will be without the online redologs. The clone of a database can also be done differently, the database can be brought up with the same SID name and a simple recovery using the online redologs can be done. After the recovery the controlfile can be recreated and all the names can be changed.

Once the backup datafiles are copied over to the clone server, verify that the permissions are set correctly. The username and group of the datafiles should be correct (This is a common problem on most sites where the Unix Admin will copy over the datafiles and forget to set the permission, losing valuable time and time spent looking out for him).

Verify that all the archive logs are in the archive directory from the time of backup until the time you want to recover. The files will have an extension of 1 or 2 depending on the

thread the archive belongs to if you have set the archive format right.

Environment Setup

Login into the primary node and verify that you are in the right environment.

Login to Node 1 (ODS034) as user oracle.

Oracle ODS034:=> id

uid=300(oracle) gid=300(dba)

Set the .profile to look for the HOST name and run the appropriate SID.env file.
Like:

```
#-------------------------------------------------------------------------------------------------------------
# If you are oracle and logging in one of the instance, then set that specific env
#-------------------------------------------------------------------------------------------------------------
if [ "`/usr/ucb/whoami`" = "oracle" ] && [ "`hostname`" = "ODS034" ]; then
. ./sidq1.env
else
. ./sidq2.env
fi
```

Check the ORACLE_SID and ORACLE_HOME (Note: the ORACLE_SID should be the name of the instance and not the database)

oracle ODS034:=> echo $ORACLE_SID

sidq1

Common Shared Oracle Home (Your choice, but common Home is easy to maintain).

oracle ODS034:=> echo $ORACLE_HOME

/sv03/sw/oracle/sidqdb/9.2.0

PFile Setup

If you have the parameter file in the $ORACLE_HOME/dbs directory, then copy over and just change the required parameters, if you have a PFILE location where you keep the parameter file (right thing to do), then you need to have a link created from the $ORACLE_HOME/dbs directory to the PFILE directory for the parameter files.

cd $PFILE

/sv03/oracle/admin/sidq/pfile/initsidq.ora

Some of the parameters that need to be changed for the clone environment are:

- **change "sidp" to "sidq"**
- **change "sidp1" to "sidq1"**
- **change "sidp2" to "sidq2"**

```
*.control_files=
(/sv00/db00/oradata/sidq/control01.ctl,/sv00/db03/oradata/sidq/control02.ctl,/sv00/db06/oradata/sidq/control03.ctl)
*.background_dump_dest='/sv03/oracle/admin/sidq/bdump'     # common for both instances
sidq1.compatible='9.2.0'
sidq2.compatible='9.2.0.4.0'   # For OLAP we need to set it up like this to avoid errors.
*.core_dump_dest='/sv03/oracle/admin/sidq/cdump'
*.db_name='sidq'
sidq1.instance_name='sidq1'
sidq2.instance_name='sidq2'
sidq1.instance_number=1
sidq2.instance_number=2
```

sidq1.local_listener="(address=(PROTOCOL=TCP)(**HOST**=ODS034.cnf.com)(PORT=1527))" # For Register
sidq2.local_listener="(address=(PROTOCOL=TCP)(**HOST**=ODS035.cnf.com)(PORT=1527))" # For Register
sidq2.remote_listener='listener_sidq1' # For listener load balance
sidq1.remote_listener='listener_sidq2'
*.log_archive_dest='/sv04/data/arch/**sidq**' # Single archive destination for all instances
sidq1.thread=1 # Undo thread for first instance
sidq2.thread=2 # Undo thread number for the second instance
sidq1.undo_tablespace='undotbs1' # Undo tablespace for first instance
sidq2.undo_tablespace='undotbs2'
*.user_dump_dest='/sv03/oracle/admin/**sidq**/udump' # common for both isnatnces

Reduce the SGA and other memory structure sizes. (You don't need a large SGA for dev and test, right!)

*.bitmap_merge_area_size=4194304
*.create_bitmap_area_size=8388608
*.hash_area_size=8421376
*.java_pool_size=150331648
*.large_pool_size=8388608
*.max_dump_file_size='2048'
*.shared_pool_size=419430400
*.sort_area_retained_size=4210688
*.sort_area_size=4210688

Drop old sidp links in the $ORACLE_HOME/dbs directory and create the three links for the clone database sidq.

Note: Keep all the names in lower characters, otherwise it all gets messed up and RAC parameters are case sensitive.

- Ln –s /sv03/oracle/admin/sidq/pfile/initsidq.ora initsidq.ora
- Ln –s /sv03/oracle/admin/sidq/pfile/initsidq1.ora initsidq1.ora
- Ln –s /sv03/oracle/admin/sidq/pfile/initsidq2.ora initsidq2.ora

The initsidq1.ora and initsidq2.ora will have just the **ifile**, a pointer to the main initsidq.ora. Like:

ifile=$ORACLE_HOME/dbs/initsidq.ora

Verify instance

You can straight away verify if your setup is right by bringing up a dummy instance.

oracle ODS034:=> sqlplus /nolog

SQL*Plus: Release 9.2.0.3.0 - Production on Thu May 22 09:16:14 2003

Copyright (c) 1982, 2002, Oracle Corporation. All rights reserved.

SQL> connect / as sysdba

Connected.

SQL> select * from v$instance;

```
INSTANCE_NUMBER INSTANCE_NAME
--------------- ----------------
HOST_NAME
----------------------------------------------------------------
VERSION           STARTUP_T STATUS       PAR    THREAD# ARCHIVE LOG_SWITCH_
----------------- --------- ------------ --- ---------- ------- -----------
LOGINS     SHU DATABASE_STATUS   INSTANCE_ROLE      ACTIVE_ST
---------- --- ----------------- ------------------ ----------
           1 sidq1
ODS098
9.2.0.3.0         07-MAY-03 OPEN         YES         2 STOPPED
ALLOWED    NO  ACTIVE           PRIMARY_INSTANCE   NORMAL
```

Create the Controlfile

Login to production database (SIDP1 or SIDP2) and:

```
Sql> connect / as sysdba
Sql> alter database backup controlfile to trace;
Sql> exit

oracle ODS044:=> cd $UDUMP
```

cp the trace file as control_sidp.sql

ftp the file to clone server where your clone database sidq resides.

Edit the file and create two files out of the controlfile, one to create the controlfile and the other portion to recreate the tempfiles, remove the recover portion of the controlfile and do it manually, you can have better control over the recovery. Like:

sidq_control.sql
sidq_recover_tempfiles.sql

Edit the controlfile sidq_control.sql and

- set the database to "sidq"
- change noresetlogs to resetlogs.
- you need to remove the redo logfiles for thread 2. You need to know which group belongs to the second thread. (Unfortunately the controlfile dump does not tell you that). In my case from Group 5 onwards the redolog files are thread two.
- The group two redolog files need to be split from the first group and added at the bottom of the control file as shown for thread 2.
- Enabling thread two should happen after the recovery. You can then drop it if you are going to stick on to a single instance.

```
STARTUP NOMOUNT
CREATE CONTROLFILE SET DATABASE "SIDQ" RESETLOGS  ARCHIVELOG
--  SET STANDBY TO MAXIMIZE PERFORMANCE
    MAXLOGFILES 32
    MAXLOGMEMBERS 2
    MAXDATAFILES 1022
    MAXINSTANCES 8
    MAXLOGHISTORY 10906
LOGFILE
  GROUP 1 '/sv00/db01/oradata/sidq/redo01.log'  SIZE 100M,
  GROUP 2 '/sv00/db02/oradata/sidq/redo02.log'  SIZE 100M,
  GROUP 3 '/sv00/db03/oradata/sidq/redo03.log'  SIZE 100M,
  GROUP 4 '/sv00/db04/oradata/sidq/redo04.log'  SIZE 100M        -- Thread 2 moved to bottom of the controlfile
  -- STANDBY LOGFILE
DATAFILE
  '/sv00/db00/oradata/sidq/system01.dbf',
  '/sv00/db13/oradata/sidq/odshd00l_137.dbf',
  '/sv00/db04/oradata/sidq/odsd00m_06.dbf',
  '/sv00/db04/oradata/sidq/odsd00m_07.dbf',
  '/sv00/db09/oradata/sidq/odsx00m_08.dbf',
'/sv00/db13/oradata/sidq/odshd00l_132.dbf',
  '/sv00/db13/oradata/sidq/odshd00l_133.dbf'
CHARACTER SET WE8ISO8859P1
;
alter database add logfile thread 2
  GROUP 5 '/sv00/db01/oradata/sidq/redo09.log'  SIZE 100M,
  GROUP 6 '/sv00/db01/oradata/sidq/redo10.log'  SIZE 100M,
  GROUP 7 '/sv00/db01/oradata/sidq/redo11.log'  SIZE 100M,
  GROUP 8 '/sv00/db01/oradata/sidq/redo12.log'  SIZE 100M,
                :
```

```
SQL> @sidq_control.sql
ORACLE instance started.
```

Total System Global Area 6147778512 bytes
Fixed Size 739280 bytes
Variable Size 771751936 bytes
Database Buffers 5368709120 bytes
Redo Buffers 6578176 bytes

Control file created.

Cancel Based Recovery

In the other portion of the sidq_recover.sql, remove the recover database commands and fire it manually.

SQL> recover database using backup controlfile until cancel;
ORA-00279: change 1937810322614 generated at 08/16/2003 12:29:16 needed for
thread 1
ORA-00289: suggestion : /sv04/data/arch/sidq/arch703203_1.arc
ORA-00280: change 1937810322614 for thread 1 is in sequence #703203

Specify log: {<RET>=suggested | filename | AUTO | CANCEL}

ORA-00279: change 1937810322614 generated at 08/16/2003 12:29:16 needed for
thread 2

It does not suggest what archivelog file to apply for thread 2. You need to choose the latest one and start applies from there.

Specify log: {<RET>=suggested | filename | AUTO | CANCEL}
/sv04/data/arch/sidq/arch3636_2.arc
ORA-00279: change 1937810325033 generated at 08/16/2003 12:35:50 needed for
thread 1
ORA-00289: suggestion : /sv04/data/arch/sidq/arch703204_1.arc
ORA-00280: change 1937810325033 for thread 1 is in sequence #703204
ORA-00278: log file '/sv04/data/arch/sidq/arch703203_1.arc' no longer needed
for this recovery

Specify log: {<RET>=suggested | filename | AUTO | CANCEL}

This time it suggests what file is required for the thread 2. The first time seems to be an issue, once it applies the first right thread 2 archivelog file, and then it prompts for more.

ORA-00279: change 1937810464267 generated at 08/16/2003 13:07:38 needed for
thread 2
ORA-00289: suggestion : /sv04/data/arch/sidq/arch3637_2.arc
ORA-00280: change 1937810464267 for thread 2 is in sequence #3637
ORA-00278: log file '/sv04/data/arch/sidq/arch3636_2.arc' no longer needed for
this recovery

Specify log: {<RET>=suggested | filename | AUTO | CANCEL}

ORA-00279: change 1937810500912 generated at 08/16/2003 13:12:27 needed for
thread 2
ORA-00289: suggestion : /sv04/data/arch/sidq/arch3638_2.arc
ORA-00280: change 1937810500912 for thread 2 is in sequence #3638
ORA-00278: log file '/sv04/data/arch/sidq/arch3637_2.arc' no longer needed for
this recovery

Specify log: {<RET>=suggested | filename | AUTO | CANCEL}

ORA-00279: change 1937810531734 generated at 08/16/2003 13:15:44 needed for
thread 2
ORA-00289: suggestion : /sv04/data/arch/sidq/arch3639_2.arc
ORA-00280: change 1937810531734 for thread 2 is in sequence #3639
ORA-00278: log file '/sv04/data/arch/sidq/arch3638_2.arc' no longer needed for
this recovery

```
Specify log: {<RET>=suggested | filename | AUTO | CANCEL}

CANCEL
Media recovery cancelled.

SQL> alter database open resetlogs;

Database altered.
```

When you do a cancel based recovery, you need to open resetlogs.

Create the tempfiles

We now add the tempfiles for the temporary tablespace, which is not part of the backup set as tempfiles do not play any role in recovery. However, if you have shadow copied the database, then the tempfiles will be part of the backup and you could still add the same.

```
Sql>@sidq_recover_tempfiles.sql
```

The file will have the below information:

```
ALTER TABLESPACE TEMP ADD TEMPFILE '/sv00/db13/oradata/sidq/temp_01.dbf'
    SIZE 2044M REUSE AUTOEXTEND OFF;
ALTER TABLESPACE TEMP ADD TEMPFILE '/sv00/db13/oradata/sidq/temp_02.dbf'
    SIZE 2044M REUSE AUTOEXTEND OFF;
ALTER TABLESPACE TEMP ADD TEMPFILE '/sv00/db13/oradata/sidq/temp_01.dbf'
    SIZE 2044M REUSE AUTOEXTEND OFF;
ALTER TABLESPACE TEMP ADD TEMPFILE '/sv00/db13/oradata/sidq/temp_02.dbf'
    SIZE 2044M REUSE AUTOEXTEND OFF;
ALTER DATABASE ENABLE THREAD 2;
```

Switch to NoArchivelog Mode

To change a database from Archivelog mode to NoArchivelog mode, you need to switch the Cluster Database option off and then change the database from Archivelog to NoArchivelog. Once the database NoArchivelog mode is switched ON, then we can shutdown the database, switch the cluster option ON and start the database again.

1. **Shutdown the database**

You can shutdown the database using the sqlplus utility or if you have setup the srvconfig already, then you can use that to shut it down as shown below.

```
ODS034=:> srvctl stop database –d sidq
```

2. **Switch the Cluster off**

You need to tell the instance that you are not part of the cluster before you can turn the archivelog mode to noarchive. Open the initSIDQ.ora file and change the parameter:

```
#*.cluster_database=TRUE
*.cluster_database=FALSE
```

3. **Start the database**

Do not use the SRVCTL utility to start the database now. Login using sqlplus, mount the database and change the database to NoArchivelog mode.

```
SQL> startup mount

ORACLE instance started.
```

```
Total System Global Area    6063892312 bytes
Fixed Size                  739160 bytes
Variable Size               687865856 bytes
Database Buffers            5368709120 bytes
Redo Buffers                6578176 bytes

SQL> select name from v$database;

NAME
---------
SIDQ

SQL> archive log list

Database log mode           Archive Mode
Automatic archival          Disabled
Archive destination         /shared/arch/oradata/sidq/arch
Oldest online log sequence  7
Current log sequence        8

SQL> alter database Noarchivelog;

Database altered.

SQL> archive log list

Database log mode           No  Archive Mode
Automatic archival          Disabled
Archive destination         /shared/arch/oradata/sidq/arch
Oldest online log sequence  7
Next log sequence to archive  8
Current log sequence        8
```

4. Shutdown the database

Shut down the database in normal mode. The bounce will take affect, the switch back of the cluster option to true and archivelog option to false.

5. Change the init parameter back

Open the initDRAC.ora file and change the parameter:

```
*.cluster_database=TRUE
#*.cluster_database=FALSE
```

Note: If you are using srvctl utility and the cloning is for more than a single node, then you need to set this parameter to TRUE, before you start the database using the srvctl utility. If you don't, then the error you get is:

```
oracle ODS034:=> srvctl start database -d sidq
PRKP-1003 : Startup operation partially failed
ORA-01102: cannot mount database in EXCLUSIVE mode
6578176 bytes
4
Copyright (c) 1982, 2002, Oracle Corporation.  All rights reserved.
```

6. Turn the Archivelog parameter to false

```
#*.log_archive_start=TRUE
*.log_archive_start=FALSE
```

7. **Startup the database in single instance mode**

```
oracle ODS034:=> srvctl start database -d sidq –I sidq1
```

Your database is now a single instance clone of production and ready to be opened to the users.

House Keeping

After you clone the database there are some house keeping jobs that needs to be done, basically these will be all pointers to production and loop back stuff that needs to be verified and dropped.

Oracle Transparent Gateway

If you have OTG installed to access DB2 databases, then the dictionary information will have the cached information of production database connection, these needs to be rest. So to reset the OTG related dictionary cache information. Login to the database and

```
Connect / as sysdba

@$ORACLE_HOME/rdbms/admin/catnohs.sql   -- This will drop the OTG dict.

@$ORACLE_HOME/rdbms/admin/caths.sql     -- This will reinstall it.
```

Loop Back Database links

```
$ Connect / as sysdba

sql>select * from dba_db_links where db_link = 'SIDQ1.CNF.COM';
```

If any loop back database link exists, like SIDQ1.CNF.COM, connect as the owner of the database link and drop it. This is because the database link that exists will become a loopback database link when the global_name is changed; it needs to be dropped now before you change the global name of the database.

Change the Global Name

```
SQL> select * from global_name;

GLOBAL_NAME
--------------------------------------------------------------------------------
SIDX.CNF.COM


SQL> alter database rename global_name to sidq;

Database altered.

SQL>  select * from global_name;

GLOBAL_NAME
--------------------------------------------------------------------------------
SIDQ.CNF.COM
```

Job Queue Processes

Any jobs that were scheduled to run in production will start running in the cloned database if the job is not dropped or broken by the concerned schema owners explicitly. To mitigate such an eventuality, you can turn off the job queue processes.

```
SQL> show parameter job

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
job_queue_processes                  integer     10
```

You can turn this off dynamically and then figure our whether to brake the job or drop the job from the job queue.

SQL> Alter system set parameter job_queue_processes = 0;

SQL> show parameter job

NAME	TYPE	VALUE
job_queue_processes	integer	0

Drop database links

Drop the database links that are not required and create the ones that are required to point to production. You should be able to automate these using scripts and there is a script of mine on RevealNet to do the same.

Change Password

Change any production schema passwords and the password of sys and system. Apart from the core oracle database DBA related schema password, you could spool all the users and change their password to something that they know will be the one they should expect whenever a clone/refresh of the dev and test instance happens. Let's say we reset the entire user's password to 'oracle'. They could then change this at their convenience.

We can write a small script to spool the output and execute the output to achieve the same.

```
Set linesize 200
Set pagesize 200
Set echo off
Spool change_passwords.sql
select 'alter user '||username||' identified by oracle;' from dba_users order by username
/
spool off
@change_passwords.sql
```

We could then change the core schema passwords of different modules to some standard password that the leads of different modules are in agreement with. You could maintain these passwords in hidden password file (this is something a lot of developers want to know on how to avoid the use of hard coded passwords in their batch jobs) and call it using a shell script and change the core schema passwords in the database. A simple script will look like:

```
#!/bin/ksh
#-----------------------------------------------------------------------------------------------
set -x
PASSFILE='/dba/etc/.dba_dba2.pwd'
cat $PASSFILE | awk -F"/" '{print $1}' | while read NAME
do
PASSWORD=`cat $PASSFILE | grep ^"$NAME/" | awk -F"/" '{print $2}`
echo 'alter user '$NAME 'identified by '$PASSWORD';' >> change_password.sql
done
#-----------------------------------------------------------------------------------------------
```

Reset iSqlPlus

To reset iSqlPlus, you need to change some parameters in two config files, one in the sqlplus home directory and the other in the Apache home directory. You also need the httpd server up and running to connect using iSqlPlus.

At a minimum you need to # out the following lines in the /sv03/sw/oracle/sidqdb/9.2.0/sqlplus/admin/isqlplus.conf files for isqlplus to start.

```
<Location /isqlplus>
  SetHandler fastcgi-script
  Order deny,allow
```

```
# Comment "Allow ..." and uncomment the four lines "AuthType ..."
# to "Require ..." if Oracle HTTP authentication access is required
# for the http://.../isqlplus URL
  Allow from all
#  AuthType Basic
#  AuthName 'iSQL*Plus'
#  AuthUserFile /sv03/sw/oracle/sidqdb/9.2.0/sqlplus/admin/iplus.pw
#  Require valid-user
</Location>
```

You need to change the production related hostnames and ports to a test or dev parameter and then start up the httpd server for iSqlPlus to connect.

Cd to /sv03/sw/oracle/sidqdb/9.2.0/Apache/Apache/conf directory and
Open the httpd.conf file. Then change the following parameters:

```
ServerName ODS044            -- Change the host name
Allow from localhost ODS044  -- Change the host name
ProcNode ODS044 7777         -- Change the host name and leave the port number alone
ProcNode ODS044 80           -- Change the host name and leave the port number alone
```

Once the changes are made you can check if there are any syntax errors using command:

```
oracle ODS034:=> apachectl configtest
Syntax OK
```

Then start the httpd server.

```
oracle ODS034:=> apachectl start
```

```
oracle ODS034:=> ps –ef | grep Apache
```

Change Listener and TNSNAMES

Copy over the listener and tnsnames.ora files to your cloned server $TNS_ADMIN directory and change the SID and Host parameters. You can leave the post numbers alone if this server is dedicated to the cloned database.

```
listener_sidq1 =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = TCP)(HOST = ODS034.cnf.com)(PORT = 1527))
      )
    )
  )
SID_LIST_listener_sidq1 =
  (SID_LIST =
    (SID_DESC =
      (ORACLE_HOME = /sv03/sw/oracle/sidqdb/9.2.0)
      (SID_NAME = sidq1)
    )
  )
```

Note: If you do not setup the listener name correctly like shown above then you end up with a PRKP-1011 oracle error when you try to start the database using the SRVCTL utility. The SRVCTL utility starts the database and the listener using the gsd daemon, but does not shut the listener down when you use it to shut down the database. You need to manually shut the listener down.

```
ODS034:=> srvctl start database -d sidq

PRKP-1011 : Failed to get all the listeners associated with instance SIDQ1 on nodeODS034
```

The tnsnames.ora file has the load balance parameter setup to load balance between the listeners.If you have just one listener setup as it's a single instance, then just bring that listener up and leave the tnsnames.ora file alone, you can use the same tnsnames.ora file and don't have to copy it over all the client machines or sites, when you decide to add a second node.

```
sidq.cnf.com =
  (DESCRIPTION =
    (LOAD_BALANCE = yes)
        (ADDRESS = (PROTOCOL = TCP)(HOST = ODS034.cnf.com)(PORT = 1527))
        (ADDRESS = (PROTOCOL = TCP)(HOST = ODS035.cnf.com)(PORT = 1527))
        (CONNECT_DATA = (SERVICE_NAME = sidq.cnf.com)))
listener_sidq1.cnf.com =(ADDRESS = (PROTOCOL = TCP)(HOST = ODS034.cnf.c om)(PORT = 1527))
listener_sidq2.cnf.com =(ADDRESS = (PROTOCOL = TCP)(HOST = ODS035.cnf.com)(PORT = 1527))
```

The listener portion of the tnsnames if for registration of the listeners when the instances come up, you will see the same parameter in the init parameter file. When the instance comes up it automatically registers with these listeners as the port numbers are not 1521 to register automatically.

Note: Like I said before, you cannot shutdown the listeners using the SRVCTL utility, but you can use the LSNRCTL utility to shutdown the listeners of any node from any node.

Reset Autoextend of datafiles

Rest all the datafiles to an autoextend off or to a size that is acceptable to your environment, if you leave the database with an autoextend on, the development instance will run away with the filesystem space that has been allocated to your database.

```
select file_name,to_char(bytes),to_char(MAXBYTES),AUTOEXTENSIBLE
from dba_data_files
where AUTOEXTENSIBLE = 'YES'
order by tablespace_name
/
```

sql> alter database datafile '/sv00/db03/oradata/rpt1/usersx05.dbf' autoextend off;

Shankar Govindan works as a Sr. Oracle DBA at CNF Inc, Portland, Oregon. Shankar Govindan is Oracle Certified 7, 8 and 8I; you can contact him at shankargovindan@yahoo.com. Note: The above info as usual is of my individual tests and opinions and has nothing to do with the company I work for or represent.