# Database Rolling Upgrade Using Transient Logical Standby Oracle Database 10g Release 2

*Oracle Maximum Availability Architecture White Paper*
*August, 2010*

# Maximum Availability Architecture

## Oracle Best Practices for High Availability

ORACLE

# Database Rolling Upgrade Using Transient Logical Standby
# Oracle Database 10g Release 2


Executive Summary ..... 2
Overview: Rolling Database Upgrade with Transient Logical Standby ..... 3
Considerations when using Transient Logical Standby ..... 5
Patch/upgrade preparation best practices ..... 5
Patch Apply/Upgrade Best Practices ..... 7
Post Patch Apply/Upgrade Best Practices ..... 8
General Restrictions ..... 8
  Transient Logical Standby Restrictions ..... 9
Fallback Best Practices ..... 9
  Backups ..... 9
  Flashback Database ..... 9
  Downgrade ..... 10
  Transient Logical Fallback Considerations ..... 11
Appendix A - Detailed Example using a Transient Logical Standby ..... 13
  High-level Steps ..... 13
  Detailed Steps ..... 14
Appendix B - Canceling a Logical Standby Switchover ..... 24
Appendix C - Cloning ORACLE_HOME for a Patchset Apply ..... 24
References ..... 25

# Database Rolling Upgrade Using Transient Logical Standby
# Oracle Database 10g Release 2

## EXECUTIVE SUMMARY

The customary process for utilizing SQL Apply to execute a database rolling upgrade is described in the Data Guard Documentation and the MAA paper, "SQL Apply Rolling Upgrade Best Practices"[3].

This paper provides an alternative to the customary process for users who prefer to use a physical standby database for day-to-day operations and who are not able to create a second standby database (a logical standby), to execute a rolling upgrade.

The procedure discussed in this paper

1. Converts an existing physical standby to a logical standby database
2. Executes a rolling upgrade
3. Reverts the standby back to its original status as a physical standby database once the upgrade is complete.

This is called a rolling upgrade using a *transient logical standby database* and is supported with Oracle Database 10g Release 2 (10.2.0.1 and onward). This paper only applies if you are starting with any release prior to 11g (11.1.0.6). If you are upgrading beginning with Oracle Database 11g, please see: Database Rolling Upgrades for Physical Standby Databases using Transient Logical Standby 11g.

## OVERVIEW: ROLLING DATABASE UPGRADE WITH TRANSIENT LOGICAL STANDBY

During a rolling upgrade using a transient logical standby database, the following high-level steps are executed:

1. Prerequisites
2. Convert the physical Standby to a Logical Standby
3. Perform the Upgrade on the Standby
4. Switchover
5. Create a Temporary Archive redo log repository (optional)
6. Convert Back to the Original Physical Standby Configuration
7. Patch the Primary Site RDBMS
8. Switch back to the original config (optional)
9. Raise the COMPATIBLE Parameter Setting

This procedure is similar to the standard SQL Apply Rolling Upgrade with the following differences:

- A physical standby control file is created on the primary for purposes of flashing back the original primary to become a physical standby post-switchover. If the target release is 11.1.0.7 or higher, then creation of a physical standby control file on the primary is not necessary.
- A guaranteed restore point is created on the original primary for purposes of flashing back to become a physical standby post-switchover.
- Converting the physical standby to a logical standby uses a manual method to maintain the database ID (V$DATABASE:DBID).
- The original primary is actually upgraded via the redo apply process after it has been converted from logical standby to being a physical standby.

Just like the customary SQL Apply Rolling Upgrade process, a logical standby database (in this paper the original physical standby database is temporarily converted to be a logical standby) is upgraded to release n+1 while production runs on the primary at release n. When the standby upgrade is deemed successful, Data Guard resynchronizes the standby with the primary, and a switchover is executed to transition the standby database to the production role running on the new release. While the standby database operates in the production role, the database on the original primary is converted back to a physical standby by flashing back the database to the guaranteed restore point and then restoring the physical standby control file. Once it regains its status as a physical standby, it is then restarted with version n+1 and upgraded to release n+1 via Redo Apply. If

desired, a second switchover can be executed to return all databases to their original role. Total database downtime is limited to the time it takes to execute a Data Guard switchover, compared to the longer downtime required for a conventional database upgrade.

The improved availability achieved by using this process versus the conventional upgrade method derives from the fact that the actual database upgrade is performed using the Database Upgrade Assistant (dbua) or the manual upgrade method (scripts, catupgrd.sql and utlrp.sql), on the logical standby without any impact to the primary database. The total outage time for a rolling upgrade using a transient logical standby is the time that it takes to complete a database switchover and the application or client reconnection time.

Proof-of-concept tests at a large Oracle customer demonstrated that it is possible to complete a database rolling upgrade with just 1 minute of database downtime (a total of 2 minutes was required to include a second switchover used to return both databases to their original role). In the same proof of concept a conventional upgrade method required over three hours of database downtime to complete. Note that of the 3 hours of total database downtime – approximately 1 hour was attributed to human error, not uncommon during the course of an upgrade where the primary database is unavailable until all upgrade tasks are completed. If this same human error had occurred in the course of a rolling upgrade using a transient logical standby, there would have been no impact on the availability of the primary database. Using a logical standby to execute the upgrade resulted in a 99% reduction in planned down time.

Oracle internal testing with the database supporting Oracle's enterprise email application showed a 96% reduction in downtime, from 48 minutes to 2 minutes, when comparing the conventional upgrade method to the SQL Apply rolling upgrade method.

This paper describes MAA best practices for using SQL Apply to perform:

- A rolling upgrade for users of physical standby, where the physical standby is temporarily converted to a logical for the duration of the upgrade, and returned to a physical standby when the upgrade is complete. This process allows physical standby users to obtain the same minimal downtime benefit of logical standby for rolling database upgrades, without requiring a second set of storage for an "extra" logical standby database, and without requiring that physical standby users switch to logical standby to do a rolling database upgrade. This process is called a rolling upgrade using a "transient physical standby".
- Fallback options in the event that a rolling database upgrade using transient logical standby has to be aborted.

The generic procedure for executing a SQL Apply Rolling Upgrade is detailed in Chapter 11 of the Data Guard Concepts & Administration guide [5]. Chapter 11 assumes the reader also has knowledge of the generic upgrade steps for any Oracle database as described in the Oracle Database Upgrade Guide 10*g* Release 2 [7]. Both of these resources must be reviewed before performing a rolling upgrade using a transient logical standby as described in this paper. This paper complements Oracle documentation with best practices provided below and is not intended to duplicate information already provided in the documentation.

## CONSIDERATIONS WHEN USING TRANSIENT LOGICAL STANDBY

The complete procedure is documented in Appendix A, "Detailed Example using a Transient Logical Standby". Note that if the primary database is a Real Application Cluster (RAC) database, you must ensure that all but one instance are shut down, and the corresponding threads are disabled before initiating the switchover. Similarly, if the logical standby database is a RAC database, ensure that all instances except the one where SQL Apply is running are shut down, and the corresponding threads are disabled before initiating the switchover. You re-enable the threads and start the instances once the switchover operation has completed successfully. Although the instances are shut down, the role change will be automatically propagated to these instances when they are restarted. These steps are included in Appendix A – Detailed Example using a Transient Logical Standby

There may also be cases when your database includes data types that SQL Apply cannot support (this is never an issue for Redo Apply – physical standby). Even in these cases, however, it may still be possible to use the transient logical standby rolling upgrade procedure. The determination has to be made if there is a satisfactory way to handle the unsupported data types. Options for using the rolling upgrade when unsupported data types exist are:

1. Temporarily suspend changes to the unsupported tables for the period of time it takes to perform the upgrade procedure.
2. Perform the upgrade at a time when users will not be making changes to the unsupported tables.
3. If you cannot prevent changes to unsupported tables during the upgrade, any unsupported transactions that occur are recorded in the `DBA_LOGSTDBY_EVENTS` table on the logical standby database. After the upgrade is completed, you may use Oracle Data Pump or the Export/Import utility to import the changed tables to the upgraded databases.

## PATCH/UPGRADE PREPARATION BEST PRACTICES

Preparing properly for the upgrade will give you the necessary knowledge and confidence to complete a successful upgrade. Chapter 2 of the Database Upgrade

guide [7] has an excellent set of steps to follow. In addition to those steps the following is also recommended:

- Review Documentation resources
    - Read the Upgrade Guide [7]
    - Read the Release Notes
    - Review known issues and post-release MetaLink notes
    - Read the steps in the Data Guard Concepts & Administration guide, chapter 11 [5]
    - Review and follow the Oracle 10g Upgrade Companion in Oracle *MetaLink* Note: 466181.1. This is an important document to review because it contains a comprehensive set of steps and best practices for upgrading.
- Follow MetaLink note 300479.1 to implement prerequisites specific to your release for executing a rolling upgrade with Transient Logical Standby.
- Clone (see "Cloning ORACLE_HOME for a Patchset Apply") a new ORACLE_HOME and then apply the patchset to it. This is not mandatory when applying patch sets but it is recommended. Having a separate installation rather than directly applying the patch set to the existing installation allows for easy switching between ORACLE_HOMEs, including if a fallback is necessary. Using a new ORACLE_HOME is also termed as an out-of-place patchset apply.
- As stated in the Upgrade guide, your test plan must include performance testing. Ensure that proper performance monitoring and statistics gathering is in place prior to the upgrade so you have a baseline of comparison during the post-upgrade evaluation phase on the logical standby and after the complete upgrade.
- Create a test plan that includes the best practices outlined in the 11g Upgrade Companion in Oracle *MetaLink* Note 466181.1, and the following additional practices:
    - Upgrade testing.
    - Fallback testing: Testing the fallback methods that will be used in the event that any step fails. Various fallback methods are discussed in the "Fallback" section later in this paper.
    - Handle unsupported tables and objects: If you have any unsupported tables and have determined a method to handle them within the rolling upgrade, then ensure that this is in your test plan

- Maintain the same database COMPATIBLE setting on all databases in the Data Guard configuration until the upgrade has been completely evaluated. For example, if upgrading from 10.2.0.4 to 11.1.0.6 then COMPATIBLE should be set to 10.2.0.4.
- When identifying logical standby unsupported data types and storage attributes as described in the Data Guard Concepts and Administration guide [5] chapter 11, also read and use the Oracle Database 10*g* Release 2 Best Practices: Data Guard SQL Apply [3] MAA white paper, particularly the section, e "Confirm Data Type Support". Note that any table that contains an unsupported data type is skipped in its entirety when using SQL Apply.
- User data should not be stored in any internal schema supplied by Oracle. Any user data placed within an internal schema as displayed in DBA_LOGSTDBY_SKIP will be skipped by SQL Apply.
- If you have any unsupported tables and have determined a method to handle them within the rolling upgrade then ensure that this is in your test plan. For example, if SQL Apply is only used for the rolling upgrade, it may be possible to prevent any updates, inserts, or deletes to such tables until after the upgrade is complete.
- Test your manual procedures for logical standby switchover and switchback. It is not possible to use the Data Guard broker or Enterprise Manager to manage a SQL Apply rolling upgrade because the broker does not currently support a Data Guard configuration where primary and standby databases are at different Oracle database releases (see General Restrictions - below).
- Follow MAA best practices and enable flashback database and create guaranteed restore points so that flashback can be used for reverting the database to a primary and as a fallback option in certain cases.
- Understand the fallback restoration options for each upgrade step described in more detail later in this paper.
    - Backups
    - Flashback Database
    - Downgrade procedure

## PATCH APPLY/UPGRADE BEST PRACTICES

- Clone the existing ORACLE_HOME and apply the patchset to the cloned ORACLE_HOME. This avoids any downtime associated with the application of the patchset to the software and carries over any post-install ORACLE_HOME changes. See Appendix F, "Cloning ORACLE_HOME for a Patchset Apply".
- You can choose either the Database Upgrade Assistant (dbua) or the manual upgrade method when executing the actual database upgrade on the logical

standby during the rolling upgrade process.. The Database Upgrade Assistant (dbua) is the recommended method and it does take care of any updates to the Oracle Cluster Registry (OCR).

- Create a guaranteed restore point on the logical standby prior to applying the patchset (on both sites when prior to running dbua or catupgrd.sql). In the event of any issues with the patchset apply, this will facilitate quick restoration of the database to it's pre-upgrade state via flashback database rather than having to use the downgrade procedure which usually takes as long as the upgrade procedure. I
- Create a temporary log archive repository at your primary site following the first switchover (while you are running production at the standby location and the original primary is still at the previous release) to reduce data loss in the case of a primary database disaster. See "Create a Temporary Archive redo log repository (optional)" in Appendix A for details.

## POST PATCH APPLY/UPGRADE BEST PRACTICES

- Consider using outage time during the switchover to also update the COMPATIBLE parameter setting.
- Tune and adjust

## GENERAL RESTRICTIONS

Restrictions that generally apply to using SQL Apply for rolling database upgrades also apply to using a transient logical standby:

- The databases must not be part of a Data Guard Broker configuration. See the Data Guard Broker guide [8] for instructions on removal of databases from a broker configuration.
- The Data Guard protection mode must be set to either maximum availability or maximum performance. Query the PROTECTION_LEVEL column in the V$DATABASE view to find out the current protection mode setting.
- The LOG_ARCHIVE_DEST_n initialization parameter for the logical standby database destination must be set to OPTIONAL to ensure the primary database can proceed while the logical standby database is being upgraded.
- The COMPATIBLE initialization parameter must match the software release prior to the upgrade. That is, a rolling upgrade from release n to release n+1 requires that the COMPATIBLE initialization parameter be set to n on both the primary and standby databases. This can be updated post upgrade after all assurance tests are passed.
- During the initial switchover in the rolling upgrade process, the PREPARE operation cannot be used because the primary and standby databases are at

different Oracle releases. Instead, perform the switchover without the prepare statements as described above.

### Transient Logical Standby Restrictions

In addition to the above, the following restrictions apply specifically to using a transient logical standby:

- Transient Logical Standby procedure can only be used from Oracle Database 10g Release 2 (10.2) onward.
- You must use the Data Guard 10.1 procedure for converting a physical standby to a logical standby so that the database ID can be maintained. The example provided in Appendix C reflects this fact.
- You cannot have any bystander logical standby databases. A bystander logical standby database would be a pre-existing logical standby database in the same Data Guard configuration as the physical standby that will become the transient logical. If you have a bystander logical standby, Oracle recommends that you use it to perform a generic SQL Apply rolling database upgrade.

## FALLBACK BEST PRACTICES

The Flowchart below (Figure 1, SQL Apply Rolling Upgrade with Transient Logical Fallback Options) presents an overview of which fallback options are viable at each step for the SQL Apply Rolling Upgrade.

### Backups

Database and software backups should be taken on the primary and the standby prior to the upgrade process. The software backups should include the oraInventory directory tree. The software backups are only necessary if they have never been done and if you are applying the patchset directly to the existing ORACLE_HOME tree rather than applying it to a newly installed separate ORACLE_HOME.

### Flashback Database

Flashback Database is a method to achieve very fast point-in-time recovery and most often will be the quickest way to fall back to the previous release. Note that data from any transactions that occur after the point in time that the database is recovered to are lost. Since a standby database is not operating in a production role, Flashback Database an excellent tool for backing out the database upgrade on the logical standby before the first switchover in the SQL Apply rolling upgrade process. Using flashback database with sufficient space and creating a guaranteed

restore point immediately prior to the upgrade will be the fastest method to fallback as compared to a restore or a downgrade.

Following switchover, and after production is running on the original standby database, any flashback operation will result in data loss. Deciding whether flashback database is a preferred to downgrading the database at this point is a tradeoff between the speed at which you need to return production to the previous release, and the amount of data loss that you can tolerate.

Note: A perquisite to using flashback database to revert to the pre-upgrade version is that the COMPATIBLE database parameter must have remained at the release you are downgrading to. Remember the restriction noted above that Flashback Database cannot be used to flashback a database from 10.2.0.x to 10.1.0.x.

The steps for flashing back the database after a failed upgrade (catupgrd.sql or dbua failure)are as follows:

1. Shutdown the database
2. Startup mount the database under the new ORACLE_HOME
3. Flashback to the guaranteed restore point taken prior to the upgrade
   SQL> flashback database to restore point PRE_LOGICAL_UPGRADE;

   If this is a pre 10.2 database then you would flashback to the SCN pint captured prior to the upgrade
   `SQL> FLASHBACK DATABASE TO SCN 1500;` (Where 1500=the SCN point)
4. Shutdown the database
5. Startup mount the database under the old ORACLE_HOME
6. Open resetlogs the database
   SQL> alter database open resetlogs;

Note: For 10.1.0.3 or later releases of Oracle Database 10g Release 1 (10.1.0.x), guaranteed restore point used in the procedure above is not available. Instead, administrators must identify the current SCN from the V$DATABASE view prior to starting the upgrade and use that SCN as the flashback point.

### Downgrade

The downgrade procedure is run manually via the steps documented in Chapter 7 of the "Oracle Database Upgrade Guide" [7]. This generally takes as long as the upgrade procedure and backs out the patchset changes while maintaining any transactional changes. In cases where transactional changes have taken place since the upgrade and there is no other way to restore the data, then the downgrade procedure should be used. A perquisite to downgrading is that the COMPATIBLE database parameter must have remained at the release you are downgrading to.

### Transient Logical Fallback Considerations

The fallback practices when using a Transient Logical Standby differ from the generic SQL Apply Rolling Upgrade best practices. Fallback practices are described in Figure 1, Fallback Options when using Transient Logical Standby.

In general, ensure that the proper backups have been taken prior to invoking the rolling upgrade.

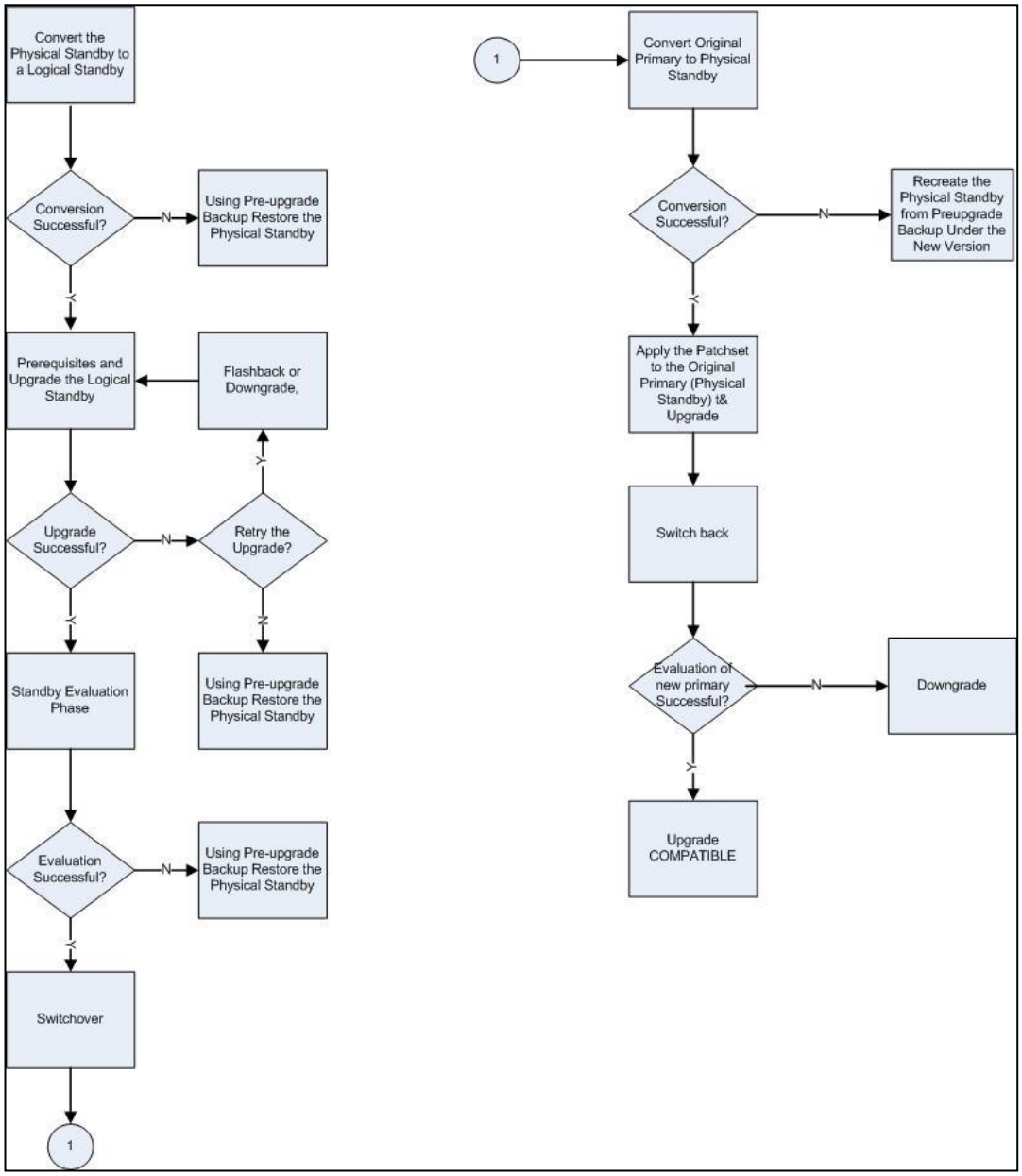


**Figure 1 Fallback Options when using Transient Logical Standby**

## APPENDIX A - DETAILED EXAMPLE USING A TRANSIENT LOGICAL STANDBY

The procedure can be summarized in the following high-level steps:

### High-level Steps

1. Prerequisites
2. Convert the physical Standby to a Logical Standby
3. Create a Temporary Archive redo log repository (optional)
4. Perform the Upgrade on the Standby
5. Switchover
6. Create a Temporary Archive redo log repository (optional)
7. Convert Back to the Original Physical Standby Configuration
8. Patch the Primary Site RDBMS
9. Switch back to the original config (optional)
10. Raise the COMPATIBLE Parameter Setting

This procedure is dependent on flashback database and also uses a modified method for converting the physical standby to a logical standby so that the database identity (v$database:DBID) can be maintained throughout the process. In the following table are the detailed set of steps when starting with a primary database and a physical standby database. These steps assume that the environments are on RAC using ASM, and Oracle Managed Files (OMF). Thee are also some other references in these steps as follows:

**Primary Site (**original primary database)
Cluster with 2 Hosts: chi01, chi02
DB_UNIQUE_NAME=chicago
ASM Disk Groups: +DATA, +RECOVERY
Oracle Managed Files being used (db_create_file_dest='+DATA' and db_recovery_file_dest='+RECOVERY')

**Standby Site** (original standby database )
Cluster with 2 Hosts: bos01, bos02
DB_UNIQUE_NAME=boston.
ASM Disk Groups: +DATA, +RECOVERY
Oracle Managed Files being used (db_create_file_dest='+DATA' and db_recovery_file_dest='+RECOVERY')

## Detailed Steps

| Step | Primary | Standby | Notes |
|---|---|---|---|
| I. | Prerequisites | | |
| 1 | Check for unsupported data types if using for upgrading a physical-only type environment.<br>Also set events to capture unsupported operations:<br>`EXEC DBMS_LOGSTDBY.APPLY_SET('MAX_EVENTS_RECORDED',DBMS_LOGSTDBY.MAX_EVENTS);`<br>`EXEC DBMS_LOGSTDBY.APPLY_SET('RECORD_UNSUPPORTED_OPERATIONS', 'TRUE');` | | See Appendix C of the Data Guard Concepts & Admin guide [5] and run the query in the MAA SQL Apply Best Practices paper in the "Confirm Data Type Support" section. Also refer to the Data Guard Administration guide [5] section 11.4, steps 2-3, to understand how to deal with any unsupported data types.<br>Make sure none of the "Restrictions" discussed earlier in this paper exist. |
| 2 | shutdown immediate<br>startup mount<br>Turn flashback database on<br>`SQL> ALTER DATABASE FLASHBACK ON;` | Cancel managed recovery<br>`SQL> RECOVER MANAGED STANDBY DATABASE CANCEL;`<br>Turn flashback database on<br>`SQL> ALTER DATABASE FLASHBACK ON;` | **Canceling managed recovery is mandatory prior to creating the logical standby control file on the primary.**<br>If flashback is already on then this can be skipped. Flashback database should be on as a fallback option in the event anything goes wrong. Initially turning on flashback database requires the database to be in MOUNT mode. |
| 3 | Create a guaranteed restore point<br>`SQL> CREATE RESTORE POINT PRE_UPGRADE GUARANTEE FLASHBACK DATABASE;` | | This will be used to flashback the primary site when it's a logical standby as part of converting back to a physical standby. It can also be used for fall back.<br>This does not require flashback database to be on, i.e. a guaranteed restore point can be created without turning flashback database on. Creating a guaranteed restore point without flashback database enabled requires a single instance to be mounted to create the initial guaranteed restore point. WE still recommend turning on flashback database for quicker fallback in the event of any unexpected outages during the process. |
| 4 | **If the target release is prior to 11.1.0.7** then create a physical standby controlfile on the primary database<br>`SQL> ALTER DATABASE CREATE PHYSICAL STANDBY CONTROLFILE AS '/home2/oracle/backup/db/ctrlphyspreupgrd.ctl' REUSE;` | | This control file will be used when converting back to a physical standby post-upgrade. This is only required if you're upgrading to a release less than 11.1.0.6 . |
| 5 | Create a new ASM directory for the logical standby archive logs on the Primary<br>Set the ASM environment and use asmcmd to create a directory in the recovery area<br>`> asmcmd`<br>`ASMCMD> cd +recovery/chicago`<br>`ASMCMD> mkdir standbyarchive`<br>If not using ASM then create a new directory | Create a new ASM directory for the logical standby archive logs on the Standby<br>Set the ASM environment and use asmcmd to create a directory in the recovery area<br>`> asmcmd`<br>`ASMCMD> cd +recovery/boston`<br>`ASMCMD> mkdir standbyarchive`<br>If not using ASM then create a new directory | This is to ensure that logical standby archived logs remain separate from SRL's that are archived. |

| Step | Primary | Standby | Notes |
|---|---|---|---|
| 6 | Set Primary Database Logical Standby Role parameters<br>`SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_3='LOCATION=+RECOVERY/chicago/standbyarchive/ VALID_FOR=(STANDBY_LOGFILES,STANDBY_ROLE) DB_UNIQUE_NAME=chicago';`<br>`SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_3=ENABLE;` | Set the Logical Standby Logfile Archive Destination on the Standby<br>`SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_3='LOCATION=+RECOVERY/boston/standbyarchive/ VALID_FOR=(STANDBY_LOGFILES,STANDBY_ROLE) DB_UNIQUE_NAME=boston';`<br>`SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_3=ENABLE;` | At this stage ensure your local archive destination has VALID_FOR=(ONLINE_LOGFILES,ALL_ROLES ) otherwise if it is set to (ALL_LOGFILES,ALL_ROLES) then you will get standby redo logs archived to both the local location and the LOG_ARCHIVE_DEST_3 location. |
| 7 | Enable supplemental logging on the Primary<br>`SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (PRIMARY KEY, UNIQUE INDEX) COLUMNS;` | | You can verify if supplemental logging is on via:<br>`SQL> select supplemental_log_data_pk,supplemental_log_data_ui from v$database;`<br><br>`SUP SUP`<br>`--- ---`<br>`YES YES` |
| 8 | Create Logical control file on the primary database<br>`SQL> ALTER DATABASE CREATE LOGICAL STANDBY CONTROLFILE AS '/home2/oracle/backup/db/ctrllogicalpreupgrd.ctl' REUSE;` | | This will be used to convert the physical standby to a logical. The logs should be transferred to the new standbyarchive location on the standby |
| II. | Convert the physical Standby to a Logical Standby (if Physical only) | | |
| 9 | | On the Standby, run SQL to Generate the RMAN Script to Rename Data Files<br>set pages 0 head off feedback off lines 150<br>set echo off termout off verify off<br>`spool RMANcatForLogiCtl.sql`<br>`select 'catalog datafilecopy ' \|\| CHR(39) \|\| name \|\| CHR(39) \|\| ';'`<br>`from v$datafile;`<br>`select 'switch database to copy;'`<br>`from dual;`<br>`spool off`<br>The script being generated will be named /home2/oracle/sql/RMANcatForLogiCtl.sql | This SQL is needed to catalog data file copies with RMAN with the correct name after using the logical standby control file from the primary.<br>The RMAN commands being generated for each data file are of the format:<br>`catalog datafilecopy '+DATA/boston/datafile/system.283.601543985';`<br>And a line at the end to switch the database to to copy is also added.<br>**If you have the exact same paths and file names (not using OMF) then this step is not necessary** |
| 10 | `Copy the logical standby controlfile from primary to standby`<br>`> /home2/oracle/backup/db`<br>`> rcp ctrlLogicalPreUpgrd.ctl bos01:`pwd`` | | |
| 11 | | Shutdown the standby database<br>`> srvctl stop database -d boston` | |
| 12 | | Replace the physical standby controlfile with the logical standby control file that was copied.<br>RMAN> STARTUP NOMOUNT;<br>RMAN> RESTORE CONTROLFILE FROM '/home2/oracle/backup/db/ctrlLogicalPreUpgrd.ctl'; | |

| Step | Primary | Standby | Notes |
|---|---|---|---|
| 13 | | Mount the standby database<br>RMAN> startup mount<br><br>Rename the data files with the RMAN script previously created<br>RMAN><br>@/home2/oracle/sql/RMANcatForLogiCtl.sql | Check for flashback logs in FRA. Possibly should disable flashback at this stage to cleanup flashback logs.<br><br>**Flashback logs are deleted at this point, verified by monitoring alert.log.**<br><br>**If you have the exact same paths and file names (not using OMF) then this step is not necessary** |
| 14 | | Check the standby log names and drop/create correct standby redo logs. Do for each group. This could be generated with the following:<br>SET PAGES 0 HEAD OFF FEEDBACK OFF LINES 150 ECHO OFF TERMOUT OFF VERIFY OFF<br>SPOOL DROPSRL.SQL<br>SELECT 'ALTER DATABASE DROP STANDBY LOGFILE GROUP ' \|\| GROUP# \|\| ';' FROM V$STANDBY_LOG;<br>SPOOL OFF;<br>Create the standby redo logs<br>(this syntax is for a database using OMF). Recommendation is: (maximum number of online logfiles for each thread + 1) * maximum number of threads<br>SQL> ALTER DATABASE ADD STANDBY LOGFILE THREAD 1 ('+RECOVERY');<br>SQL> ALTER DATABASE ADD STANDBY LOGFILE THREAD 2 ('+RECOVERY');<br><br>Repeat the SRL creates at least twice for each thread so there are 3 SRL's per thread (this assumes that there were 2 online logs per thread). | These steps assume that standby_file_management=AUTO, thus the online redo logs will be automatically created once recovery starts. |
| 15 | | Restart Managed Recovery on the standby database<br>SQL> RECOVER MANAGED STANDBY DATABASE | Monitor the alert.log. This will recover the standby to the incomplete recovery point SCN, the logical dictionary_end SCN. Should see "Incomplete Recovery" message in alert.log when this completes.<br>Incomplete Recovery applied until change 39018095 |
| 16 | | Activate the standby database<br>SQL> ALTER DATABASE ACTIVATE STANDBY DATABASE; | The database is now a logical standby |
| 17 | | Bounce the logical standby database<br>SQL> SHUTDOWN IMMEDIATE;<br>SQL> STARTUP MOUNT | |
| 18 | | Turn flashback database back on<br>SQL> ALTER DATABASE FLASHBACK ON;<br>SQL> ALTER DATABASE OPEN; | Need to re-enable flashback as it is was turned off during the conversion to a logical standby. Note that any restore points created prior to the conversion to a logical are also gone. |

| Step | Primary | Standby | Notes |
|---|---|---|---|
| 19 | | Start Logical standby apply<br>`SQL> ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE;` | |
| 20 | | Turn SQL Apply Auto Delete Off<br>`SQL> EXECUTE DBMS_LOGSTDBY.APPLY_SET('LOG_AUTO_DELETE','FALSE');` | This is a safeguard against logs needed for SQL Apply mining being deleted |
| 21 | | Verify that the logical standby is running correctly | See 9.3 Monitoring a Logical Standby Database |
| III. | Create a Temporary Archive redo log repository (optional) | | |
| 22 | | Follow MetaLink Note 434164.1 | Optionally, an archived redo log repository can be created with the same database version and COMPATIBLE setting as the primary so that redo is still received during the patch apply/upgrade. See MetaLink Note 434164.1 for setting that up. This will ensure that the recovery point objective (RPO) can be met in the event of a primary site failure during this step. |
| IV. | Perform the Upgrade on the Standby | | |
| 23 | | Create a guaranteed restore point<br>`SQL> CREATE RESTORE POINT PRE_LOGICAL_UPGRADE GUARANTEE FLASHBACK DATABASE;` | have to re-enable flashback or restore points. |
| 24 | | Stop Logical Standby Apply<br>`SQL> ALTER DATABASE STOP LOGICAL STANDBY APPLY ;` | |
| 25 | | If not using DBUA then set cluster_database=false in preparation for upgrade<br>`SQL> ALTER SYSTEM SET CLUSTER_DATABASE=FALSE SCOPE=SPFILE;` | DBUA requires leaving CLUSTER_DATABASE=TRUE for RAC databases. |
| 26 | | Shutdown the logical standby database | |
| 27 | Defer redo transfer on the primary<br>`SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=DEFER;` | | |
| 28 | | Apply 11.1.0.6 patchset to CRS and ASM | See "Optimizing Availability During Planned Maintenance Using Oracle Clusterware and RAC" |
| 29 | | Implement the workaround for Issue 5236922 | See 300479.1 for the workaround |

| Step | Primary | Standby | Notes |
|---|---|---|---|
| 30 | | Apply the 11.1.0.6 Patch set | You can now choose either the Database Upgrade Assistant (DBUA) or the manual upgrade method with catupgrd.sql when executing the actual database upgrade on the logical standby. Using the DBUA is the recommended method. It takes care of all parameter changes and of any updates to the Oracle Cluster Registry (OCR) in the case of a RAC system. Refer to the "Oracle Database Upgrade Guide 11.1 " [7] for complete upgrade instructions.<br><br>Oracle recommend using an out-of-place patchset apply. The existing ORACLE_HOME can be cloned using the Cloning procedure in the appendix. Major release upgrades have to have a separate ORACLE_HOME. |
| 31 | If a temporary log archive repository was used then register logs so they don't have to be resent:<br>`SQL> ALTER DATABASE REGISTER LOGFILE '….`<br><br>Optionally, you can use the RMAN CATALOG command to catalog the archived redo log repository archivelog destination. e.g.:<br>`RMAN> CATALOG START WITH`<br>`'+ASMDG/ALOGREP/ARCHIVELOG/';` | | This is if an archive log repository was used to cover the exposure period. If a log archive repository was used then the log register commands can be generated with a script like the following:<br>::::::::::::::<br>register_logs.sql<br>::::::::::::::<br>REM The parameter is the SCN from the 'recover standby database command<br>set head off feedback off lines 133 pages 0 verify off echo off<br>spool register_logs_for_standby.sql<br>select 'alter database register logfile ' \|\| chr(39) \|\| name \|\| chr(39) \|\| ';'<br>from v$archived_log<br>where first_change# >= &1<br>/<br>spool off |
| 32 | Enable redo transfer<br>`SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE;` | Start Logical Standby Apply @ 11.1.0.6<br>`SQL> alter database start logical standby apply immediate;` | |
| 33 | | | |
| 34 | | Verify that the logical standby is running correctly | See 9.3 Monitoring a Logical Standby Database |

| Step | Primary | Standby | Notes |
|---|---|---|---|
| V. | Switchover | | |
| 35 | Switchover to Logical Standby on current/original primary<br>SQL> SELECT SWITCHOVER_STATUS FROM V$DATABASE;<br><br>SWITCHOVER_STATUS<br>--------------------<br>TO STANDBY<br><br>If the query returns "SESSIONS ACTIVE', then ensure it's ok to shut them down.<br>SELECT SID, PROCESS, PROGRAM FROM V$SESSION WHERE TYPE = 'USER'<br>AND SID <> (SELECT DISTINCT SID FROM V$MYSTAT);<br><br>$ srvctl stop instance -d chicago -i chicago2<br>SQL> ALTER DATABASE DISABLE THREAD 2;<br>SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;<br>SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;<br>SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO LOGICAL STANDBY; | | Make sure this is aligned with MAA Switchover best practices. You cannot do an "ALTER DATABASE PREPARE TO SWITCHOVER" for this switchover due to the mixed versions. This is not supported.<br><br>"SWITCHOVER TO LOGICAL STANDBY;" cannot use the "WITH SESSION SHUTDOWN" clause so you need to manually shutdown sessions. |
| 36 | | Ensure archive logs are received and applied during the primary switchover to a logical | Monitor the alert logs |
| 37 | | Defer Redo from the Primary (original standby)<br>SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=DEFER; | |
| 38 | | If this is a RAC database and the target release is pre-11g then do the following:<br>$ srvctl stop instance -d boston -i chicago2<br>SQL> ALTER DATABASE DISABLE THREAD 2; | If the target release, which the logical standby has been upgraded to, is 11g, then it is no longer necessary to shutdown the other RAC instances nor is it necessary to disable the non-participating database instance threads. |
| 39 | | Switchover to Primary on current logical standby<br><br>SQL> SELECT SWITCHOVER_STATUS FROM V$DATABASE;<br><br>SWITCHOVER_STATUS<br>--------------------<br>TO PRIMARY<br><br>SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY; | If switchover_status does not change to 'TO PRIMARY' as required then the switchover can be backed out at this point, see Appendix D, Logical switchover backout steps.<br><br>You cannot do an "ALTER DATABASE PREPARE TO SWITCHOVER" for this switchover due to the mixed versions. |

| Step | Primary | Standby | Notes |
|---|---|---|---|
| 40 | | If this is a RAC database and the target release is pre-11g then do the following:<br>Start other nodes of the primary (original standby database)<br>`SQL> ALTER DATABASE ENABLE THREAD 2;`<br>`$ srvctl start instance -d boston -i boston2` | If the target release, which the logical standby has been upgrade to, is 11g, then it is no longer necessary to shutdown the other RAC instances nor is it necessary to disable the non-participating database instance threads. |
| 41 | | At this point the applications can be started pointing to the new upgraded primary | |
| VI. | Create a Temporary Archive redo log repository (optional) | | |
| 42 | Install a separate ORACLE_HOME and patch it to 11.1.0.6 | | Optionally, an archived redo log repository can be created with the same database version and COMPATIBLE setting as the primary so that redo is still received during the patch apply/upgrade. See MetaLink Note 434164.1 for setting that up. This will ensure that the recovery point objective (RPO) can be met in the event of a primary site failure during this step. |
| VII. | Convert Back to the Original Physical Standby Configuration (if Physical Only Environment) | | |
| 43 | Flashback the original primary database (currently a logical standby) to guaranteed restore point in the "Create a guarantee restore point on the primary" step<br>`SQL> ALTER DATABASE ENABLE THREAD 2;`<br>`SQL> SHUTDOWN IMMEDIATE;`<br>`SQL> STARTUP MOUNT`<br>`SQL> FLASHBACK DATABASE TO RESTORE POINT PRE_UPGRADE;` | | |
| 44 | Shutdown the original primary database<br>`SQL> SHUTDOWN IMMEDIATE;` | | |
| 45 | **If target release is prior to 11.1.0.7**<br>Startup nomount the primary database using the physical standby control file created in the "Create Physical Standby Control File" step<br>`RMAN> STARTUP NOMOUNT`<br>`RMAN> RESTORE CONTROLFILE FROM '/home2/oracle/backup/db/ctrlPhysPreUpgrd.ctl'` | | This is not required if the target release is 11.1.0.7 or greater since that release adds the "ALTER DATABASE CONVERT TO PHYSICAL STANDBY;" command. |
| VIII. | Patch the Primary Site RDBMS ORACLE_HOME | | |
| 46 | Shutdown the original primary database<br>`SQL> shutdown immediate;` | | |
| 47 | Apply 11.1.0.6 patchset to CRS and ASM | | See "Optimizing Availability During Planned Maintenance Using Oracle Clusterware and RAC" |
| 48 | Install the new 11.1.0.6 Patch set out-of-place | | Use an out-of-place patchset apply. The existing ORACLE_HOME can be cloned using the Cloning procedure in the appendix. |
| 49 | Startup mount the database **with the new ORACLE_HOME** environment<br>`SQL> STARTUP MOUNT` | | This will be started as a physical standby now |

| Step | Primary | Standby | Notes |
|---|---|---|---|
| 50 | **If target release is 11.1.0.7 or greater**<br><br>ALTER DATABASE CONVERT TO PHYSICAL STANDBY;<br>SHUTDOWN IMMEDIATE<br>STARTUP MOUNT; | | |
| 51 | Turn flashback database on<br>SQL> ALTER DATABASE FLASHBACK ON; | | |
| 52 | If a temporary log archive repository was used then register logs so they don't have to be resent:<br>SQL> ALTER DATABASE REGISTER LOGFILE '….<br><br>Optionally, you can use the RMAN CATALOG command to catalog the archived redo log repository archivelog destination. e.g.:<br>RMAN> CATALOG START WITH<br>'+ASMDG/ALOGREP/ARCHIVELOG/'; | | This is if an archive log repository was used to cover the exposure period. If a log archive repository was used then the log register commands can be generated with a script like the following:<br>::::::::::::::<br>register_logs.sql<br>::::::::::::::<br>REM The parameter is the SCN from the 'recover standby database command<br>set head off feedback off lines 133 pages 0 verify off echo off<br>spool register_logs_for_standby.sql<br>select 'alter database register logfile ' \|\| chr(39) \|\| name \|\| chr(39) \|\| ';'<br>from v$archived_log<br>where first_change# >= &1<br>/<br>spool off |

| | | | |
|---|---|---|---|
| 53 | | Enable Log Transport on the original standby database (currently the primary)<br>`SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE;` | **If target release is prior to 11.1.0.7**<br>**Wait for the log file to be registered on the original primary database by FAL**<br>**DO NOT PERFORM A LOG SWITCH SPECIFICALLY**<br>**Wait for all logs to be sent by monitoring the alert.log** |
| 54 | **If target release is prior to 11.1.0.7**<br>Verify that the new incarnation branch has been registered and matches the primary by running this query on each database:<br>select INCARNATION#, RESETLOGS_CHANGE#, PRIOR_RESETLOGS_CHANGE#, STATUS, RESETLOGS_ID, PRIOR_INCARNATION# from v$database_incarnation where status='CURRENT'; | **If target release is prior to 11.1.0.7**<br>Verify that the new incarnation branch has been registered and matches the primary by running this query on each database:<br>select INCARNATION#, RESETLOGS_CHANGE#, PRIOR_RESETLOGS_CHANGE#, STATUS, RESETLOGS_ID, PRIOR_INCARNATION# from v$database_incarnation where status='CURRENT'; | **If target release is prior to 11.1.0.7**<br>Review the alert.log. The registration happens automatically as the redo stream is received.. Query output from each database should be identical.<br><br>In 11.1.0.7 or greater there is an internal check made by managed recovery to verify that the new incarnation is received. Managed recovery will wait 10 seconds and check for the new incarnation until the new incarnation is received and registered. |
| 55 | **Once the above queries match**, start managed recovery on the original primary database (now a physical standby)<br>**If target release is 11.1.0.7 or greater then it is not necessary to wait or run the query to verify the new incarnation branch is registered.**<br>`SQL> RECOVER MANAGED STANDBY DATABASE DISCONNECT;` | | The above queries must match before starting managed recovery. If managed recovery is started before these queries match then you will invalidate your standby database.<br>In 11.1.0.7 and forward there is a safeguard and managed recovery will not start until the new incarnation is received.<br>This is where the upgrade will happen via the redo apply. |
| 56 | Verify that the physical standby is running correctly | | See 8.5 Monitoring the Primary and Standby Databases |
| IX. | Switch back to the original config (optional) | | |
| 57 | | Switchover to physical standby on original standby database<br>`$ srvctl stop instance -d boston -i boston2`<br>`SQL> SELECT SWITCHOVER_STATUS FROM V$DATABASE;`<br><br>`SWITCHOVER_STATUS`<br>`--------------------`<br>`TO STANDBY`<br><br>`SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO PHYSICAL STANDBY;`<br>`Or if sessions were active that can be shutdown then use:`<br>`SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO PHYSICAL STANDBY WITH SESSION SHUTDOWN;`<br>`SQL> SHUTDOWN IMMEDIATE;`<br>`SQL> STARTUP MOUNT;` | Note that the switchover will apply all redo through the upgrade as well. |

| | | | |
|---|---|---|---|
| 58 | Switchover to primary on original primary database<br>`$ srvctl stop instance -d chicago -i chicago2`<br>`SQL> SELECT SWITCHOVER_STATUS FROM V$DATABASE;`<br><br>`SWITCHOVER_STATUS`<br>`--------------------`<br>`TO PRIMARY`<br><br>`SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY;`<br>`Or if sessions were active that can be shutdown then use:`<br>`SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY WITH SESSION SHUTDOWN;`<br>`SQL> ALTER DATABASE OPEN;` | | |
| 59 | | Start Managed Recovery on the standby database<br>`SQL> RECOVER MANAGED STANDBY DATABASE DISCONNECT;` | |
| 60 | | Mount Other Instances<br>`$ srvctl start database -d boston -o mount` | |
| 61 | Start all primary database instances<br>`$ srvctl start database -d chicago` | | |
| X. | Raise the COMPATIBLE Parameter Setting | | |
| 63 | Once test results are satisfactory then the COMPATIBLE parameter setting can be raised if there new features to be used. | | Note that raising the COMPATIBLE setting requires a database restart and eliminates any ability to downgrade. |
| 64 | | `ALTER SYSTEM SET compatible='11.1.0.6' SCOPE=SPFILE;` | |
| 65 | `ALTER SYSTEM SET compatible='11.1.0.6' SCOPE=SPFILE;` | | |
| 66 | | Bounce the standby database<br>`> srvctl stop database -d boston`<br>`> srvctl start database -d boston` | |
| 67 | Bounce the primary database<br>`> srvctl stop database -d chicago`<br>`> srvctl start database -d chicago` | | |

## APPENDIX B - CANCELING A LOGICAL STANDBY SWITCHOVER

If you have completed the "ALTER DATABASE COMMIT TO SWITCHOVER TO LOGICAL STANDBY;" command on the primary and determine that the logical standby cannot switch to a primary at that point then you can back out of the switchover process as follows:

1. On the primary switch it back to a primary,
   ```
   SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO LOGICAL
   PRIMARY;
   ```
2. At this point when or if you want to restart SQL Apply on the logical standby you must use the "NEW PRIMARY" clause to reestablish the build of the Log Miner multi-versioned Data Dictionary:
   ```
   SQL> ALTER DATABASE START LOGICAL STANDBY APPLY
   IMMEDIATE NEW PRIMARY TO_CHICAGO;
   ```
   Where TO_CHICAGO is the database link from the logical standby to the primary created as part of the "Prepare Upgrade" best practices.

## APPENDIX C - CLONING ORACLE_HOME FOR A PATCHSET APPLY

For an out-of-place (creating a new ORACLE_HOME) patchset apply, it is recommended to clone the existing ORACLE_HOME using the procedure below.

1. As the root user, on each node in the cluster copy the existing ORACLE_HOME to the new ORACLE_HOME location. (This requires no downtime)

   ```
   > cp –pr  /u01/app/oracle/product/10.2.0.4
   /u01/app/oracle/product/11.1.0.6
   ```

2. As the OS user that owns the Oracle software (e.g. oracle), add the new ORACLE_HOME software to the Oracle Inventory using a cloning script.
   e.g.:
   ```
   #!/bin/sh
   echo "Clone started at `date`" | tee -a clone.log
   perl
   /u01/app/oracle/product/11.1.0.6/clone/bin/clone.pl
   ORACLE_HOME=/u01/app/oracle/product/11.1.0.6
   ORACLE_HOME_NAME=10gR2P2 '-
   O"CLUSTER_NODES={chi01,chi02}"' '-
   O"LOCAL_NODE=chi01"'
   echo "Clone ended at `date`" | tee -a clone.log
   ```
3. As root user run root.sh on each node
4. Apply the patchset to newly cloned ORACLE_HOME (/u01/app/oracle/product/11.1.0.6)

## REFERENCES


1. Oracle Data Guard
   http://www.oracle.com/technetwork/database/features/availability/dataguardoverview-098960.html
2. Oracle Maximum Availability Architecture
   http://www.oracle.com/technetwork/database/features/availability/maa-096107.html
3. Oracle Database 10g Release 2 Best Practices: Data Guard SQL Apply
   http://www.oracle.com/technetwork/database/features/availability/maa-wp-11gr1-sqlapplybestpractices-131426.pdf
4. Oracle Database 10g Release 2 Best Practices: Data Guard Switchover and Failover
   http://www.oracle.com/technetwork/database/features/availability/maa-wp-10gr2-switchoverfailoverbest-128455.pdf
5. Oracle Data Guard Concepts and Administration (Part #B14239-01)
   http://download-west.oracle.com/docs/cd/B19306_01/server.102/b14239/toc.htm
6. Oracle Database High Availability Best Practices 10g Release 2 - Documentation
   http://download.oracle.com/docs/cd/B19306_01/server.102/b25159/toc.htm
7. Oracle Database Upgrade Guide 10*g* Release 2 (10.2) (Part Number B14238-01)
   http://download-west.oracle.com/docs/cd/B19306_01/server.102/b14238/toc.htm
   Oracle Database Upgrade Guide 11.1 (Part #B28300-02)
   http://download.oracle.com/docs/cd/B28359_01/server.111/b28300/toc.htm
8. Oracle Data Guard Broker 10g Release 2 (10.2) (Part Number B14230-02)
   http://download-west.oracle.com/docs/cd/B19306_01/server.102/b14230/toc.htm
9. Oracle Database High Availability Best Practices 10g Release 2 (10.2) (Part Number B25159-01)
   http://download-west.oracle.com/docs/cd/B19306_01/server.102/b25159/toc.htm
10. Oracle Database Backup and Recovery Basics 10g Release 2 (10.2) (Part Number B14192-03)
    http://download-west.oracle.com/docs/cd/B19306_01/backup.102/b14192/toc.htm
11. MetaLink note 300479.1 , "Rolling Upgrades with Logical Standby"
12. Oracle by Example (OBE) of SQL Apply Rolling Upgrade,
    http://www.oracle.com/technology/obe/demos/admin/demos.html

**Database Rolling Upgrade Using Transient Logical Standby: Oracle Database 10g Release 2**
**August, 2010**
**Authors: Ray Dutcher**
**Contributing Authors: Lawrence To, Joe Meeks, Robert Theiler**

**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**